1.1高精度计算

主讲老师：党东

1.1高精度计算

巴蜀中学 BASHU SECONDARY SCHOOL

计算：

327 + 462 = 789

HOW?

324115125126435413425132156819385954818897
+
462231451512412415124312513412451415215125
=
786346576638847828549444670231837370034022

利用程序设计的方法去实现这样的高精度计算。

1.1高精度计算

学习目标：

- 高精度加法
- 高精度减法
- 高精度乘法
  - 高精度乘单精度
  - 高精度乘高精度
- 高精度除法
  - 高精度除单精度
  - 高精度除高精度

【例1】高精度加法 --1140

【问题描述】输入两个整数x,y，输出它们的和。

【文件输入】输入两个整数x,y（0<=x,y<=10^100）

【文件输出】输出它们的和

【样例输入】

123

234

【样例输出】357

【例1】高精度加法 --1140

问题1：高精度数的存储，用字符串读入

```
void init(int a[])
{    string s;
     cin>>s;   //读入字符串s
     a[0]=s.length();  //用a[0]计算字符串s的位数
     for(i=1;i<=a[0];i++)a[i]=s[a[0]-i]-'0';
          //将数串s转换为数组a，并倒序存储.
}
```

	s[0]	s[1]	s[2]
cin>>s //输入321	3	2	1

	a[0]	a[1]	a[2]	a[3]	a[4]
倒序储存	3	1	2	3	

【例1】高精度加法 --1140

问题2：如何高精度数相加？

观察人工计算的方法：竖式加法

```
        223
+       996
-----------
          9     3+6=9
         1      2+9=11,当前位1,进位1
        2       2+9+1=12,当前位2,进位1
      1         进位的1
-----------
      1219
```

【例1】高精度加法 --1140

问题2：如何高精度数相加?

转化为数组计算：

	a[0]	a[1]	a[2]	a[3]	a[4]
第一个数：223	3	3	2	2	

	b[0]	b[1]	b[2]	b[3]	b[4]
第一个数：996	3	6	9	9	

初始位数：max(a[0],b[0])		9	11	11	

	a[0]	a[1]	a[2]	a[3]	a[4]
两数相加：1219	4	9	1	2	1
		进位：0	进位：1	进位：1	和的位数+1

【例1】高精度加法 --1140

问题2：如何高精度数相加?

【代码】

```
void jia(int a[],int b[]) //计算a=a+b
{    int i,k;
     if(a[0]<b[0])a[0]=b[0]; //确定加法最大位数
     for(i=1;i<=a[0];i++)a[i]+=b[i]; //逐位相加
     for(i=1;i<=a[0];i++)
     {   a[i+1]+=a[i]/10;
         a[i]%=10;
     } //处理进位
     if(a[a[0]+1]>0)a[0]++;//修正a的位数(a+b最多只能进一位)
}
```

【例1】高精度加法 --1140

问题2：高精度数如何输出?

【代码】

```
void print(int a[])  //打印输出
{    int i;
      if(a[0]==0){cout<<0<<endl;return;}
      for(i=a[0];i>=1;i--)cout<<a[i];
      cout<<endl;
}
```

	a[0]	a[1]	a[2]	a[3]	a[4]
倒序输出	4	9	1	2	1

输出结果：1219

【例2】高精度减法 --1141

Description

输入两个整数x,y，输出它们的差x-y。

Input

输入两个整数x,y（0 <= x,y <= 10^100）

Output

输出它们的差

Sample Input

234

123

Sample Output

111

【例2】高精度减法 --1141

【分析】

竖式减法：

```
          927
-         896
------------
            1     7-6=1
          3       12-9=3,借位12
        0         (9-1)-8=0
------------
          31
```

【例2】高精度减法 --1141

问题1：如何比较高精度数大小?

```
int compare(int a[],int b[])
    //比较a和b的大小关系，若a>b则为1，  a<b则为-1,a=b则为0
{   int i;
    if(a[0]>b[0])return 1;   //a的位数大于b,则a比b大
    if(a[0]<b[0])return -1;  //a的位数小于b,则a比b小
    for(i=a[0];i>=1;i--)  //否则a和b的位数相同，则从高位到低位比较
    {    if(a[i]>b[i])return 1;
         if(a[i]<b[i])return -1;
    }
    return 0;//各位都相等则两数相等。
}
```

【例2】高精度减法 --1141

巴蜀中学 BASHU SECONDARY SCHOOL

问题2：高精度数如何相减?

转化为数组计算：

	a[0]	a[1]	a[2]	a[3]	a[4]
第一个数：927	3	7	2	9	

	b[0]	b[1]	b[2]	b[3]	b[4]
第一个数：896	3	6	9	8	

差的初始位数：max(a[0],b[0])

	a[0]	a[1]	a[2]	a[3]	a[4]
两数相加：31	2	1	3	0	

去掉多余的0，差的位数=2

不借位

借位：1
当前大小：12

不借位
当前大小：9-1

【例2】高精度减法 --1141

问题3：如何去掉多余的0?

转化为数组计算：

两数相加：31

a[0]	a[1]	a[2]	a[3]	a[4]
2	1	3	0	

当前位数
a[0]=2
此刻a[a[0]]=3
最高位数不等于0，
位数修正完成

初始位数
a[0]=3
此刻a[a[0]]=0
最高位数为0，位数减一
a[0]--;

代码实现

```
while(a[0]>0&&a[a[0]]==0) a[0]--; //修正a的位数
```

【例2】高精度减法 --1141

【思想】先判断大小，分情况用大数减小数的原则；

```
void jian(int a[],int b[])//计算a=a-b
{    int flag,i;
     flag=compare(a,b); //调用比较函数判断大小
     if (flag==0) {a[0]=0;return;} //相等
     if (flag==-1)                 //小于  则用a=b-a,返回-1
     {   cout<<"-";a[0]=b[0];
         for(i=1;i<=a[0];i++)swap(a[i],b[i]);
     }
     for(i=1;i<=a[0];i++)
     {    if(a[i]<b[i]){ a[i+1]--;a[i]+=10;} //若不够减则向上借一位
          a[i]=a[i]-b[i];
     }
     while(a[0]>0&&a[a[0]]==0) a[0]--; //修正a的位数
     return;
}
```

【例3】高精度乘法 --1142

问题1：高精度*单精度

竖式乘法：

1. 高精度乘单精度，比如12345*9=?

```
        12345
*           9
--------------
           45   5*9=45
          36     4*9=36
         27      3*9=27
        18       2*9=18
+       9        1*9=9
--------------
      111105
```

【例3】高精度乘法 --1142

问题1：高精度*单精度

转化为数组计算：

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
高精度数：12345	3	5	4	3	2	1	
单精度数：9				9			
		45	36	27	18	9	

初始位数：a[0]

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
乘积结果：111105	6	5	0	1	1	1	1

修正位数=6　　45进位4　　36+4进位4　　27+4进位3　　18+3进位2　　9+2进位1　　1

【例3】高精度乘法 --1142

问题1：高精度*单精度

【代码实现】

```
void chengdan(int a[],int k) //a=a*k,k是单精度数
{   int i;
    for(i=1;i<=a[0];i++)a[i]=a[i]*k;//先每位乘起来
    for(i=1;i<=a[0];i++){a[i+1]+=a[i]/10;a[i]%=10;} //处理进位
    while(a[a[0]+1]>0) //处理最高位相乘的进位
    {  a[0]++;
       a[a[0]+1]=a[a[0]]/10;
       a[a[0]]=a[a[0]]%10;
    }
}
```

【例3】高精度乘法 --1143

问题2：高精度*高精度

竖式乘法：

2.高精度乘高精度，比如12345*12345=?

问题转化为若干个高精度乘单精度之和

即：

先高精度*单精度

再依次高精度+高精度

【例3】高精度乘法 --1143

Description

输入两个整数x,y，输出它们的积。

Input

输入两个整数x,y（0 <= x,y <= 10^100)

Output

输出它们的积

Sample Input

```
11
12
```

Sample Output

```
132
```

【例3】高精度乘法 --1143

【代码】

```
void chenggao(int a[],int b[],int c[])
{    int i,j,len;
     for(i=1;i<=a[0];i++)
           for(j=1;j<=b[0];j++)  c[i+j-1]+=a[i]*b[j];
     c[0]=a[0]+b[0];    //位数最多为两个高精度为数之和
     for(i=1;i<=c[0];i++){c[i+1]+=c[i]/10;c[i]%=10;} //处理进位
     while(c[0]>0&&c[c[0]]==0)c[0]--;
}
```

【例4】高精度除法 --1144

问题1：高精度除单精度

竖式除法

```
              36
           ________
12345 √ 452678

        37035        12345*3=37035

         8232        45267-37035=8232
        ______
         82328       下一位再次减法

         74070       12345*6=74070
        ______
          8258       82328-74070=8258 余数
```

【例4】高精度除法 --1144

问题1：高精度/单精度

转化为数组计算：

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
高精度数：452678	6	8	7	6	2	5	4

单精度数：12345

12345

(8+8232*10)/12345=6
(8+8232*10)%12345=8258

(6+452*10)/12345=0
(6+452*10)%12345=4526

(5+4*10)/12345=0
(5+4*10)%12345=45

初始位数：a[0]

	c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]
除法结果：36	2	6	3	0	0	0	0

去掉多余的0，修正位数=2

(7+4526*10)/12345=3
(7+4526*10)%12345=8232

(2+45*10)/12345=0
(2+45*10)%12345=452

4/12345=0
4%12345=4

【例4】高精度除法 --1144

Description

输入两个整数x,y，输出它们的商。

Input

输入两个整数x,y（0 <= x <= 10^100，y <= 30000)

Output

输出它们的商

Sample Input

```
123
12
```

Sample Output

```
10
```

【例4】高精度除法 --1144

【代码】

```
void chudan(int a[],int b,int c[],int d)//商c=a/b,余数d=a%b
{    int i;
     d=0; //余数初始化
     for(i=a[0];i>=1;i--)  //按照由高位到低位的顺序,逐位相除
     {    d=d*10+a[i];   //接受了来自第i+1位的余数
          c[i]=d/b;        //计算商的第i位
          d=d%b;          //计算第i位的余数
     }
     c[0]=a[0];
     while(c[0]>0&&c[c[0]]==0)c[0]--;//计算商的有效位数
}
```

【例4】高精度除法 --1144

【优化代码】

```
void chudan(int a[],int b)
{     int i,d=0;
      for(i=a[0];i>=1;i--)//按照由高位到底位的顺序,逐位相除
      {   d=d*10+a[i]; //接受了来自第i+1位的余数
          a[i]=d/b;    //计算商的第i位
          d=d%b;       //计算第i位的余数
      }
      while(a[0]>0&&a[a[0]]==0)a[0]--;//计算商的有效位数
}
```

【练习1】求n! --1146

Description

精确计算n的阶乘n!（1 <= n < 350）

Input

输入n

Output

输出n的阶乘的值

Sample Input

3

Sample Output

6

【练习2】回文数（NOIP1999普及） --1219

【问题描述】

若一个数（首位不为零）从左向右读与从右向左读都是一样，我们就将其称之为回文数。例如：给定一个 10进制数 56，将 56加 65（即把56从右向左读），得到121是一个回文数。

又如，对于10进制数87：

STEPl:87 + 78=165；　　STEP2:165 + 561= 726；

STEP3:726 + 627 = 1353；　STEP4:1353+3531=4884

在这里的一步是指进行了一次N进制的加法，上例最少用了4步得到回文数4884。

写一个程序，给定一个N(2<n<=16）进制数 M．求最少经过几步可以得到回文数。如果在30步以内（包含30步）不可能得到回文数，则输出“Impossible”

【样例输入】9 87

【样例输出】6

【练习3】奇怪的贸易 --1466

【问题描述】

刚结束了CS战斗的小D又进入了EVE的游戏世界,在游戏中小D是一名商人,每天要做的事情就是在这里买东西,再运到那里去卖.这次小D来到了陌生的X星,X星上有n种货物,小D决定每种都买走一些,他用ai来表示第i种货物购买的数量,X星人对物品的单价有特别的决定方式.他们首先会选择一个基本价x,第一种物品单价为x,第二种物品单价为x^2,第三种物品单价为x^3……第i种物品单价为x^i.结算总价时,你还需要给他们一笔手续费a0,小D不知道自己带的钱是否能够进行这笔交易,所以请你帮助他计算这笔交易他要支付的总金额是多少.

【输入文件】

第一行两个数分别表示基准价x (x<=10),物品种数n (n<=100000)

第二行一个数,手续费a0 (a0<=100)

接下来的n行每行一个数,第i行表示第i种物品购买的数量(ai<=100)

【输出文件】输出结果的最后100位,若不足100位请高位用零补足

【样例输入】
```
2 3
4
3
2
1
```

【样例输出】
```
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000026
```

【练习4】【2003普及】麦森数 --1205

【问题描述】

形如2^P-1的素数称为麦森数，这时P一定也是个素数。但反过来不一定，即如果P是个素数，2^P-1不一定也是素数。到1998年底，人们已找到了37个麦森数。最大的一个是P=3021377，它有909526位。麦森数有许多重要应用，它与完全数密切相关。

任务：从文件中输入P（1000<P<3100000），计算2^P-1的位数和最后500位数字（用十进制高精度数表示）

【输入文件】

文件中只包含一个整数P（1000<P<3100000）

【输出文件】

第一行：十进制高精度数2^P-1的位数。

第2-11行：十进制高精度数2^P-1的最后500位数字。（每行输出50位，共输出10行，不足500位时高位补0）

不必验证2^P-1与P是否为素数。

【练习4】【2003普及】麦森数 --1205

【问题描述】

形如2^P-1的素数称为麦森数，这时P一定也是个素数。但反过来不一定，即如果P是个素数，2^P-1不一定也是素数。到1998年底，人们已找到了37个麦森数。最大的一个是P=3021377，它有909526位。麦森数有许多重要应用，它与完全数密切相关。

任务：从文件中输入P（1000<P<3100000），计算2^P-1的位数和最后500位数字（用十进制高精度数表示）

【样例输入】1279

【样例输出】

386
00000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000
00000000000000104079321946643990819252403273640855
38615262247266704805319112350403608059673360298012
23944173232418484242161395428100779138356624832346
49081399066056773207629241295093892203457731833496
61583550472959420547689811211693677147548478866962
50138443826029173234888531116082853841658502825560
46662248318909188018470682222031405210266984354887
32958028878050869736186900714720710555703168729087