
中国科学技术大学
University of Science and Technology of China
程序设计 II
Programming Design II
高精度计算
主讲：吴锋

目录
CONTENTS

例题1：大整数加法
例题2：大整数减法
例题3：大整数乘法
例题4：大整数除法
例题5：循环数

例题1：大整数加法

- 问题描述（P144）：求两个不超过200位的非负整数的和。
- 输入数据：
  - 有两行，每行是一个不超过200位的非负整数，没有多余的前导0。
- 输出要求：
  - 一行，即相加后的结果。结果里不能有多余的前导0，即如果结果是342，那么就不能输出为0342。
- 输入样例
  - 22222222222222222
  - 33333333333333333
- 输出样例
  - 55555555555555555555

◎ 例题1：大整数加法

- 解题思路
  - 用字符型或整型数组来存放大整数。
    - an[0]存放个位数
    - an[1]存放十位数
    - an[2]存放百位数
    - ……
  - 模拟小学生列竖式做加法，从个位开始逐位相加，超过或达到10则进位。
    - 用unsigned an1[201]保存第一个数，用unsigned an2[200]表示第二个数
    - 然后逐位相加，相加的结果直接存放在an1中，要注意处理进位

例题1：大整数加法

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 201
int an1[MAX_LEN+10], an2[MAX_LEN+10];
char szLine1[MAX_LEN+10], szLine2[MAX_LEN+10];
//将长度最多为 nMaxLen 的大整数 an1和an2 相加，结果放在an1, an1[0],an2[0]对应于个位
int Add(int nMaxLen , int * an1, int * an2) {
     int nHighestPos = 0;
     for(int i = 0; i < nMaxLen; i++) {
          an1[i] += an2[i];                    //逐位相加
          if(an1[i] >= 10) {                   //看是否要进位
               an1[i] -= 10;an1[i+1]++;        //进位
          }
          if(an1[i]) nHighestPos = i; //记录最高位的位置
     }
     return nHighestPos;
}
```

例题1：大整数加法

```
int main()   {
      scanf("%s", szLine1);  scanf("%s", szLine2);
      //库函数memeset将地址an1开始的sizeof(an1)字节内容置成0
      memset(an1, 0, sizeof(an1));
      memset(an2, 0, sizeof(an2));
      //下面将szLine1中存储的字符串形式的整数转换到an1中去，an1[0]对应于个位
      int nLen1 = strlen(szLine1);
      for(int j =0, i = nLen1 - 1;i >= 0 ; i--)
            an1[j++] = szLine1[i] - '0';
      int nLen2 = strlen(szLine2);
      for(int j=0, i = nLen2 - 1; i >= 0; i--)
            an2[j++] = szLine2[i] - '0';
      int nHighestPos = Add(MAX_LEN,an1,an2);
      for(int i = nHighestPos; i >= 0; i--)  printf("%d", an1[i]);
      return 0;
}
```

例题2：大整数减法

- 问题描述：求2个大的正整数相减的差。
- 输入数据
  - 第1行是测试数据的组数n，每组测试数据占2行，第1行是被减数a，第2行是减数b(a > b)。
  - 每组测试数据之间有一个空行，每行数据不超过100个字符。
- 输出要求
  - n行，每组测试数据有一行输出是相应的整数差。

例题2：大整数减法

- 输入样例
  - 2
  - 99999999999999999999999999999999999999
  - 9999999999999
  - 54096567750978508956870567980689709345465465756767686784354353 45
  - 1
- 输出样例
  - 99999999999999999999990000000000000000
  - 54096567750978508956870567980689709345465465756767686784354353 44

例题2：大整数减法

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 110
int an1[MAX_LEN], an2[MAX_LEN];
char szLine1[MAX_LEN], szLine2[MAX_LEN];
//两个最多nMaxLen位的大整数an1减去an2, an1保证大于an2
int Substract(int nMaxLen, int * an1, int * an2) {
    int nStartPos = 0;
    for(int i = 0; i < nMaxLen; i++) {
        an1[i] -= an2[i];                //逐位减
        if(an1[i] < 0) {                 //看是否要借位
            an1[i] += 10; an1[i+1]--; //借位
        }
        if(an1[i]) nStartPos = i;        //记录最高位的位置
    }
    return nStartPos; //返回值是结果里面最高位的位置
}
```

例题2：大整数减法

```
int main() {
      int n;
      scanf("%d", &n);
      while(n--) {
            scanf("%s", szLine1); scanf("%s", szLine2);
            memset(an1, 0, sizeof(an1)); memset(an2, 0, sizeof(an2));
            //下面将szLine1中存储的字符串形式的整数转换到an1中去，an1[0]对应于个位
            int nLen1 = strlen( szLine1);
            for(int j = 0, i = nLen1 - 1; i >= 0; i--)an1[j++] = szLine1[i] - '0';
            int nLen2 = strlen(szLine2);
            for(int j = 0, i = nLen2 - 1; i >= 0; i--)an2[j++] = szLine2[i] - '0';
            int nStartPos = Substract(MAX_LEN, an1,an2);
            for(int i = nStartPos; i >= 0; i--) printf("%d", an1[i]);
            printf("\n");
      }
     return 0;
}
```

例题3：大整数乘法

- 问题描述（P146）：求两个不超过200位的非负整数的积。
- 输入数据
  - 有两行，每行是一个不超过200位的非负整数，没有多余的前导0。
- 输出要求
  - 一行，即相乘后的结果。结果里不能有多余的前导0，即如果结果是342，那么就不能输出为0342。
- 输入样例
  - 12345678900
  - 98765432100
- 输出样例
  - 1219326311126352690000

例题3：大整数乘法

- 解题思路
  - 用unsigned an1[200]和unsigned an2[200]分别存放两个乘数，用aResult[400]来存放积。计算的中间结果也都存在aResult中。aResult长度取400是因为两个200位的数相乘，积最多会有400位。an1[0], an2[0], aResult[0]都表示个位。
  - 一个数的第i位和另一个数的第j位相乘所得的数，一定是要累加到结果的第i+j位上。这里i, j都是从右往左，从0开始数。
  - 计算的过程基本上和小学生列竖式做乘法相同。为编程方便，并不急于处理进位，而将进位问题留待最后统一处理。

例题3：大整数乘法

- 现以 835×49为例来说明程序的计算过程：
  - 先算835×9。5×9得到45个1，3×9得到27个10，8×9得到72个100。由于不急于处理进位，所以835×9算完后，aResult如下：

下标		5	4	3	2	1	0
aResult	····	0	0	0	72	27	45

  - 接下来算4×5。此处4×5的结果代表20个10，因此要 aResult[1]+=20，变为：

下标		5	4	3	2	1	0
aResult	····		0	0	72	**47**	45

例题3：大整数乘法

- 现以 835×49为例来说明程序的计算过程：（续）
  - 再下来算4×3。此处4×3的结果代表12个100，因此要 aResult[2]+= 12，变为：

下标		5	4	3	2	1	0
aResult	····	0	0	0	**84**	47	45

  - 最后算 4×8。此处4×8的结果代表 32个1000，因此要aResult[3]+= 32，变为：

下标		5	4	3	2	1	0
aResult	····	0	0	**32**	84	47	45

例题3：大整数乘法

- 现以 835×49为例来说明程序的计算过程：（续）
  - 乘法过程完毕。接下来从 aResult[0]开始向高位逐位处理进位问题。aResult[0]留下5，把4加到aResult[1]上，aResult[1]变为51后，应留下1，把5加到aResult[2]上……最终使得aResult里的每个元素都是1位数，结果就算出来了：

下标		5	4	3	2	1	0
aResult	····	0	4	0	9	1	5

例题3：大整数乘法

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 200
unsigned an1[MAX_LEN+10], an2[MAX_LEN+10], aResult[MAX_LEN * 2 + 10];
char szLine1[MAX_LEN+10], szLine2[MAX_LEN+10];
int Multiply(int nLen1, int nLen2) {
      //每一轮都用an1的一位，去和an2各位相乘，从an1的个位开始
     for(int i = 0;i < nLen2; i++)  { //用选定的an1的那一位，去乘an2的各位
          for(int j = 0; j < nLen1; j++) //两数第i, j位相乘，累加到结果的第i+j位
               aResult[i+j] += an2[i]*an1[j];
     }
     //下面的循环统一处理进位问题
     int nHighestPos = 0;
     for(int i = 0; i < MAX_LEN * 2; i++) {
          if(aResult[i] >= 10) {
               aResult[i+1] += aResult[i] / 10; aResult[i] %= 10;
          }
          if(aResult[i]) nHighestPos = i;
     }
    return nHighestPos;
}
```

例题3：大整数乘法

```
int main() {
    gets_s(szLine1, MAX_LEN+10); gets_s(szLine2, MAX_LEN+10); //gets_s函数读取一行
    memset(an1, 0, sizeof(an1)); memset( an2, 0, sizeof(an2));
    memset(aResult, 0, sizeof(aResult));
    int nLen1 = strlen(szLine1);
     for(int j = 0, i = nLen1 - 1; i >= 0; i--)
          an1[j++] = szLine1[i] - '0';
     int nLen2 = strlen(szLine2);
     for(int j = 0, i = nLen2 - 1; i >= 0; i--)
          an2[j++] = szLine2[i] - '0';
     int nHighestPos = Multiply(nLen1, nLen2);
     for(int i = nHighestPos; i >= 0; i--)
          printf("%d", aResult[i]);
     return 0;
}
```

例题4：大整数除法

- 问题描述（P149）
  - 求2个大的正整数相除的商
- 输入数据
  - 第1行是测试数据的组数n，每组测试数据占2行，第1行是被除数，第2行是除数。每组测试数据之间有一个空行，每行数据不超过100个字符。
- 输出数据
  - n行，每组测试数据有一行输出是相应的整数商。

例题4：大整数除法

- 输入样例
  - 3
  - 240533731296337335900926045774205743923049649393035559579766079108273964 6
  - 298719258531870175258442993116087037290707924897109501250979055088379319789 4
  - 10000000000000000000000000000000000000000
  - 10000000000
  - 5409656775097850895687056798068970934546546575676768678435435345
  - 1
- 输出样例
  - 0
  - 10000000000000000000000000000000
  - 5409656775097850895687056798068970934546546575676768678435435345

◎ 例题4：大整数除法

- 解题思路
  - 基本的思想是反复做减法，看看从被除数里最多能减去多少个除数，商就是多少。
  - 一个一个减显然太慢，如何减得更快一些呢？以7546除以23为例来看一下：开始商为0。先减去23的100倍，就是2300，发现够减3次，余下646。于是商的值就增加300。然后用646减去230，发现够减2次，余下186，于是商的值增加20。最后用186减去23，够减8次，因此最终商就是328。
  - 所以本题的核心是要写一个大整数的减法函数，然后反复调用该函数进行减法操作。

例题4：大整数除法

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 110
int an1[MAX_LEN], an2[MAX_LEN], tmpAn2[MAX_LEN], anResult[MAX_LEN];
char szLine1[MAX_LEN], szLine2[MAX_LEN], szNouse[MAX_LEN];
int main() {
      int n;
      scanf("%d", &n);
      while(n--) {
            gets_s(szLine1, MAX_LEN); gets_s(szLine2, MAX_LEN); gets_S(szNouse, MAX_LEN);
            memset(an1, 0, sizeof(an1)); memset(an2, 0, sizeof(an2));
            //下面将szLine1中存储的字符串形式的整数转换到an1中去，an1[0]对应于个位
            int nLen1 = strlen(szLine1);
            for(int j = 0, i = nLen1 - 1; i >= 0; i--)
                  an1[j++] = szLine1[i] - '0';
            int nLen2 = strlen(szLine2);
            for(int j = 0, i = nLen2 - 1; i >= 0; i--)
                  an2[j++] = szLine2[i] - '0';
            ... ...
            printf("\n");
      }
     return 0;
}
```

例题4：大整数除法

```
int Length( int nMaxLen,int * an);
int * Max(int nMaxLen, int * an1, int * an2);
void ShiftLeft( int nMaxLen,int * an1, int * an2, int n);
int Substract( int nMaxLen, int * an1, int * an2);
int main() {
      ... ...
            int nHighestPos = 0;
            memset(anResult,0,sizeof(anResult));
            int nShiftLen = Length(MAX_LEN,an1) - Length(MAX_LEN,an2);
            while(Max(MAX_LEN,an1,an2) == an1) { //只要an1大于an2，就不停相减
                  ShiftLeft(MAX_LEN, an2, tmpAn2,nShiftLen); //算出an1的10的nShiftLen次方倍
                  //重复减去an1的10的nShiftLen次方倍，看能减几次
                  while( Max(MAX_LEN,an1,tmpAn2) == an1) {
                        Substract(MAX_LEN, an1,tmpAn2);
                        anResult[nShiftLen]++; //记录商对应位
                  }
                  if(nHighestPos == 0 && anResult[nShiftLen])
                        nHighestPos = nShiftLen; //记录结果最高位的位置
                  nShiftLen --;
            }
            for(int i = nHighestPos; i >= 0; i--) printf("%d", anResult[i]);
      ... ...
}
```

例题4：大整数除法

```
//求大整数的位数。0 算 0 位
int Length( int nMaxLen,int * an) {
     for(int i = nMaxLen-1; an[i] == 0 && i >= 0; i--);
     if(i >= 0) return i + 1; else return 0;
}

//求大整数an1和an2里面大的那个，如果都是0，则返回NULL
int * Max(int nMaxLen, int * an1, int * an2) {
     bool bBothZero = true;
     for(int i = nMaxLen -1; i >= 0; i--) {
          if(an1[i] > an2[i]) return an1;
          else if(an1[i] < an2[i]) return an2;
          else if(an1[i]) bBothZero = false;
     }
     if(bBothZero) return NULL; else return an1;
}
```

例题4：大整数除法

```
//将大整数an1左移n位，即乘以10的n次方，结果放到an2里
void ShiftLeft(int nMaxLen,int * an1, int * an2, int n) {
      memcpy(an2, an1, nMaxLen * sizeof(int));
      if(n <= 0) return;
      for(int i = nMaxLen -1 ; i >= 0; i--)
            if(i - n >= 0) an2[i] = an1[i-n];
            else an2[i] = 0;
}

//大整数an1减去an2。两者最多 nMaxLen 位，an1必须不小于an2，差放在an1里
//返回差的最高非0位的位置
int Substract( int nMaxLen, int * an1, int * an2) {
    int nStartPos = 0;
     for(int i = 0;i < nMaxLen ; i++) {
            an1[i] -= an2[i];                  //逐位相
            if(an1[i] < 0) {                   //看是否要进位
                  an1[i] += 10; an1[i+1] --;//进位
            }
            if(an1[i]) nStartPos = i;          //记录最高位的位置
      }
      return nStartPos;
}
```

例题5：循环数

- 问题描述
  - 当一个N位的整数X满足下列条件时，称其为循环数：X与任意一个整数1≤Y≤N相乘时，都将产生一个X的“循环”。即：分别将这两个整数的第1位数字与最后1位数字连在一起，可以得到一个相同的数字循环；当然两个整数在该数字循环中的起始位置不同。
  - 例如，142857是一个循环数，则：

142857 * 1 = 142857
142857 * 2 = 285714
142857 * 3 = 428571
142857 * 4 = 571428
142857 * 5 = 714285
142857 * 6 = 857142

例题5：循环数

- 输入
  - 写一个程序判断一个整数是否是循环数。输入文件是一个整数序列，每个整数长度为2~60。注意：每个整数前面的零被看作是该整数的一部分，在计算N时要统计。例如“01”是一个2位的整数，而“1”是一个1位的整数。
- 输出
  - 对每个输入整数，输出一行，说明该整数是否是循环数。

例题5：循环数

- 样例输入

  142857

  142856

  142858

  01

  058823529411764 7

- 样例输出

  142857 is cyclic

  142856 is not cyclic

  142858 is not cyclic

  01 is not cyclic

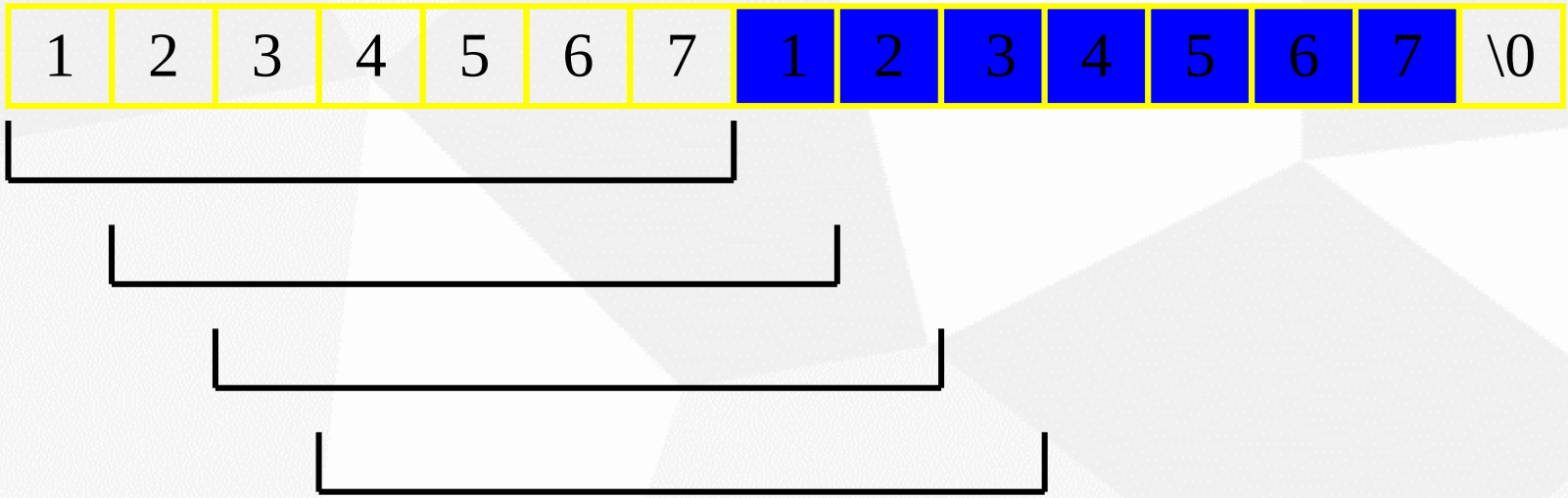
  058823529411764 7 is cyclic

例题5：循环数

- 解题思路
  - 高精度的乘法：整数可能达60位
    - X*1: $Y_0 = X$;
    - X*2: $Y_1 = Y_0 + X$
    - X*3: $Y_2 = Y_1 + X$
    - ……
  - Y_i是否是X的“循环”？

例题5：循环数

- 解题思路
  - Y_i是否是X的“循环”？
    - 穷举：N位整数，循环移位可以有N种可能
    - 循环移位方法：Y_i是否是“XX”的子串？



例题5：循环数

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 201
int an1[MAX_LEN+10], an2[MAX_LEN+10];
char szLine1[MAX_LEN+10], szLine2[MAX_LEN+10], szDouble[2 * MAX_LEN + 10];
//将长度最多为 nMaxLen 的大整数 an1和an2 相加，结果放在an1, an1[0],an2[0]对应于个位
int Add(int nMaxLen, int * an1, int * an2) {
      int nHighestPos = 0;
      for(int i = 0; i < nMaxLen; i++) {
            an1[i] += an2[i];                  //逐位相加
            if(an1[i] >= 10) {                 //看是否要进位
                  an1[i] -= 10; an1[i+1]++; //进位
            }
            if(an1[i]) nHighestPos = i; //记录最高位的位置
      }
      return nHighestPos;
}
```

例题5：循环数

```
int main() {
     while(gets_s(szLine1, MAX_LEN+10) && szLine1[0]) {
          memset(an1, 0, sizeof(an1)); memset(an2, 0, sizeof(an1));
         //下面将szLine1中存储的字符串形式的整数转换到an1中去，an1[0]对应于个位
          int nLen1 = strlen( szLine1);
          for(int j = 0, i = nLen1 - 1; i >= 0; i--) {
               an1[j] = szLine1[i] - '0'; an2[j++] = szLine1[i] - '0';
          }
          strcpy(szDouble, szLine1); strcat(szDouble, szLine1);int bOk = 1;
          for(int i = 1; i < nLen1; i++) {
               int nHighestPos = Add(MAX_LEN,an1,an2);
               if (nHighestPos >= nLen1 ) { //长度超出了
                    bOk = 0; break;
               }
               for(int k = 0; k < nLen1; k ++ )  //转换成字符串
                    szLine2[nLen1- k - 1] = an1[k] + '0';
               szLine2[nLen1] = 0;
               if(strstr(szDouble, szLine2 ) == NULL) {bOk = 0; break;}
          }
          if(bOk) printf("%s is cyclic\n", szLine1);
          else printf("%s is not cyclic\n", szLine1);
     }
    return 0;
}
```

作业

- 1. 计算2 的N 次方（P156）
  - 任意给定一个正整数N(N<=100)，计算2 的N 次方的值。
- 2. 浮点数加法（P156）
  - 求2 个不超过100 位的浮点数相加的和
- 3. 孙子问题浮点数加法（P156）
  - 对于给定的正整数a1, a2, ... an，问是否存在正整数b1, b2, ... bn，使得对于任意的一个正整数N，如果用N 除以a1 的余数是p1，用N 除以a2 的余数是p2……用N 除以an 的余数是pn，那么M = p1 * b1 + p2 * b2 + ... + pn * bn 能满足M 除以a1 的余数也是p1，M 除以a2 的余数也是p2……M 除以an 的余数也是pn。如果存在，则输出b1, b2, ... bn。题中1 <= n <= 10，a1, a2, ... an 均不大于50。
- 4. 浮点数求高精度幂（P156）
  - 有一个实数 R (0.0 < R < 99.999) , 要求写程序精确计算 R 的 n 次方。n 是整数并且 0 < n <= 25。