

```

1  #include <stdio.h>
2  #include <malloc.h>
3  #include <assert.h>
4
5  struct Node
6  {
7      int digit;
8      struct Node *next, *prev;
9  };
10 struct UBigNumber
11 {
12     int digitCount;
13     struct Node *pHead, *pTail;
14 };
15
16 struct UBigNumber InputUBN ();
17 void PrintUBN (struct UBigNumber ubn);
18 struct UBigNumber AddUBN (struct UBigNumber *pA, struct UBigNumber *pB);
19 void DestoryUBN (struct UBigNumber *pA);
20 void _InitUBN (struct UBigNumber *pUBN);
21 void _AppendDigit (struct UBigNumber *pUBN, int digit);
22 void _AppendFrontDigit (struct UBigNumber *pUBN, int digit);
23 void _Normalize (struct UBigNumber *pUBN);
24 struct Node *_NewNode ();
25 struct UBigNumber MinusUBN(struct UBigNumber *pA, struct UBigNumber *pB);
26 int Compare(struct UBigNumber *pA, struct UBigNumber *pB);//1代表前面大, -1代
    表后面大
27 struct UBigNumber MinusBN (struct UBigNumber *pA, struct UBigNumber
    *pB);//有符号大数减
28 struct UBigNumber AddBN (struct UBigNumber *pA, struct UBigNumber *pB);//有
    符号大数加
29 struct UBigNumber Multi(struct UBigNumber *pA, struct UBigNumber *pB);//无
    符号大数相乘
30 struct UBigNumber MultiOne(struct UBigNumber* pA,int x);//大数乘一个一位数
31 struct UBigNumber _Sub(struct UBigNumber *pA,int start,int end);//大数相乘中
    的切割工具
32 struct UBigNumber Multiply(struct UBigNumber *pA, struct UBigNumber
    *pB);//有符号大数相乘
33
34
35 int main ()
36 {
37     struct UBigNumber A, B, C,D;
38     A = InputUBN ();
39     B = InputUBN ();
40     /*    C = AddUBN (&A, &B);
41         D = MinusUBN(&A, &B);
42
43         PrintUBN (A);
44         printf (" + ");
45         PrintUBN (B);
46         printf (" = ");
47         PrintUBN (C);
48         printf("\n");
49         PrintUBN (A);
50         printf (" - ");

```

```

50     PrintUBN (B);
51     printf (" = ");
52     PrintUBN (D);    */
53     /*int com = Compare(&A,&B);
54     if(com==1){
55         printf("A>B\n");
56     }else if(com== -1){
57         printf("A<B\n");
58     }else{
59         printf("A=B\n");
60     }
61     PrintUBN (A);
62     printf (" + ");
63     PrintUBN (B);
64     printf (" = ");*/
65     PrintUBN (AddBN(&A,&B));
66     printf("\n");
67     /*PrintUBN (A);
68     printf (" - ");
69     PrintUBN (B);
70     printf (" = ");*/
71     PrintUBN (MinusBN(&A,&B));
72     printf("\n");
73     /*PrintUBN (A);
74     printf (" * ");
75     PrintUBN (B);
76     printf (" = ");*/
77     PrintUBN(Multiply(&A,&B));
78     DestoryUBN (&A);
79     DestoryUBN (&B);
80     //DestoryUBN (&C);
81     return 0;
82 }
83 struct UBigNumber Multiply(struct UBigNumber *pA, struct UBigNumber *pB){
84     if((pA->pHead->digit==1 && pB->pHead->digit==1) || (pA->pHead->
>digit==0 && pB->pHead->digit==0)){//同号得正
85         struct UBigNumber result = Multi(pA,pB);
86         result.pHead->digit=0;
87         return result;
88     }else {
89         struct UBigNumber result = Multi(pA,pB);
90         result.pHead->digit=1;
91         return result;
92     }
93 }
94 struct UBigNumber Multi(struct UBigNumber *pA, struct UBigNumber *pB){
95     //递归终止条件
96     if(pB->digitCount==1){
97         return MultiOne(pA,pB->pTail->digit);
98     }else if(pA->digitCount==1){
99         return MultiOne(pB,pA->pTail->digit);
100     }
101     int m=pA->digitCount;
102     int n=pB->digitCount;
103     int h=(m>n?m:n)/2;
104     struct UBigNumber a,b,c,d;
105     a=_Sub(pA,0,m-h);
106     b=_Sub(pA,m-h,m);

```

```

107     c=_Sub(pB,0,n-h);
108     d=_Sub(pB,n-h,n);
109
110     struct UBigNumber z0,z1,z2;
111     z2=Multi(&a,&c);
112     z0=Multi(&b,&d);
113     struct UBigNumber t1,t2,t3,t4,t5,result;
114     t1=AddUBN(&a,&b);
115     t2=AddUBN(&c,&d);
116
117
118     DestoryUBN(&a);
119     DestoryUBN(&b);
120     DestoryUBN(&c);
121     DestoryUBN(&d);
122
123
124     t3=Multi(&t1,&t2);
125     t4=AddUBN(&z0,&z2);
126     z1=MinusUBN(&t3,&t4);
127
128     int i;
129     for(i=0;i<2*h;i++){
130         _AppendDigit(&z2,0);
131     }
132     for(i=0;i<h;i++){
133         _AppendDigit(&z1,0);
134     }
135     t5=AddUBN(&z2,&z1);
136     result=AddUBN(&t5,&z0);
137     DestoryUBN(&z0);
138     DestoryUBN(&z1);
139     DestoryUBN(&z2);
140     DestoryUBN(&t1);
141     DestoryUBN(&t2);
142     DestoryUBN(&t3);
143     DestoryUBN(&t4);
144     DestoryUBN(&t5);
145
146     return result;
147
148 }
149 struct UBigNumber MultiOne(struct UBigNumber* pA,int x){//一个大数乘1位数
150     struct UBigNumber result;
151     _InitUBN(&result);
152     if(x==0){
153         _AppendDigit(&result,0);
154         return result;
155     }
156     int flag=0;
157     struct Node* p;
158     p=pA->pTail;
159     while(p!=pA->pHead){
160         int digit=p->digit*x+flag;
161         flag=digit/10;
162         digit%=10;
163         _AppendFrontDigit(&result,digit);
164         p=p->prev;

```

```

165     }
166     if(flag!=0){
167         _AppendFrontDigit(&result,flag);
168     }
169     return result;
170
171
172 }
173 struct UBigNumber _Sub(struct UBigNumber *pA,int start,int end){//切割, 返回
    大数中[start,end)数字子序列组成的无符号大数, 超出部分为0
174
175     struct UBigNumber result;
176     _InitUBN(&result);
177     int i=0;
178     struct Node* p=pA->pHead->next;
179     while(i<start && p!=NULL){
180         p=p->next;
181         i++;
182     }
183     while(i<end && p!=NULL){
184         _AppendDigit(&result,p->digit);
185         p=p->next;
186         i++;
187     }
188     _Normalize(&result);
189     return result;
190
191 }
192 int Compare(struct UBigNumber *pA, struct UBigNumber *pB){
193     if(pA->pHead->digit==1 && pB->pHead->digit==0){//前 负, 后 正
194         return -1;//后面大
195     }else if(pA->pHead->digit==0 && pB->pHead->digit==1){
196         return 1;//前面大
197     }else if(pA->pHead->digit==0 && pB->pHead->digit==0){
198         //先比较位数, 多者 大
199         int a=0,b=0;
200         struct Node *p1,*p2;
201         p1=pA->pHead->next;
202         p2=pB->pHead->next;
203         while(p1!=pA->pTail){
204             a++;
205             p1=p1->next;
206         }
207         a++;
208         while(p2!=pB->pTail){
209             b++;
210             p2=p2->next;
211         }
212         b++;
213         if(a>b){
214             return 1;
215         }else if(a<b){
216             return -1;
217         }else{//位数相等时, 从头开始比, 先大者大
218             p1=pA->pHead->next;
219             p2=pB->pHead->next;
220             while(p1!=pA->pTail && p2!=pB->pTail){
221                 if(p1->digit > p2->digit){

```

```

222         return 1;
223     }else if(p1->digit < p2->digit){
224         return -1;
225     }else {
226         p1=p1->next;
227         p2=p2->next;
228     }
229 }
230 }
231 return 0;//两者相等
232 }
233 }else{//都为负的情况，为都为正的情况符合相反,1改为-1，-1改为1
234     //先比较位数，多者 小
235     int a=0,b=0;
236     struct Node *p1,*p2;
237     p1=pA->pHead->next;
238     p2=pB->pHead->next;
239     while(p1!=pA->pTail){
240         a++;
241         p1=p1->next;
242     }
243     a++;
244     while(p2!=pB->pTail){
245         b++;
246         p2=p2->next;
247     }
248     b++;
249     if(a>b){
250         return -1;
251     }else if(a<b){
252         return 1;
253     }else{//位数相等时，从头开始比，先大者小
254         p1=pA->pHead->next;
255         p2=pB->pHead->next;
256         while(p1!=pA->pTail && p2!=pB->pTail){
257             if(p1->digit > p2->digit){
258                 return -1;
259             }else if(p1->digit < p2->digit){
260                 return 1;
261             }else {
262                 p1=p1->next;
263                 p2=p2->next;
264             }
265         }
266     }
267     return 0;//两者相等
268 }
269 }
270 }
271
272 struct UBigNumber InputUBN ()
273 {
274     struct UBigNumber result;
275     _InitUBN(&result);
276
277     char ch;
278     do
279         ch = getchar ();

```

```

280     while ((ch < '0' || ch > '9') && ch!='-');
281     if(ch=='-'){
282         result.pHead->digit=1;//1代表是负数
283         ch = getchar ();
284     }
285     while (ch >= '0' && ch <= '9')
286     {
287         _AppendDigit (&result, ch - '0');
288         ch = getchar ();
289     }
290     _Normalize(&result);
291     return result;
292 }
293 void PrintUBN (struct UBigNumber ubn)
294 {
295     assert (ubn.digitCount > 0 && ubn.pHead->next != NULL);
296     struct Node *la = ubn.pHead->next;
297     if(ubn.pHead->digit==1){
298         printf("-");
299     }
300     while (la)
301     {
302         printf ("%d", la->digit);
303         la = la->next;
304     }
305 }
306 struct UBigNumber MinusUBN (struct UBigNumber *pA, struct UBigNumber *pB)
307 {
308     struct UBigNumber result, *pResult = &result;
309     int flag=0;
310     _InitUBN(pResult);
311     struct Node *p1, *p2;
312     p1 = pA->pTail;
313     p2 = pB->pTail;
314     while (p1 != pA->pHead && p2 != pB->pHead)
315     {
316         int digit = p1->digit - p2->digit + flag;
317         flag=0;
318         if(digit>=0){
319             _AppendFrontDigit (pResult, digit);
320             p1 = p1->prev;
321             p2 = p2->prev;
322         }else{
323             _AppendFrontDigit (pResult, digit+10);
324             p1 = p1->prev;
325             p2 = p2->prev;
326             flag=-1;
327         }
328     }
329     while (p1 != pA->pHead->next && p1 != pA->pHead)
330     {
331         int digit = (p1->digit) + flag;
332         flag=0;
333         if(digit<0){
334             _AppendFrontDigit (pResult, digit+10);
335             p1 = p1->prev;
336             flag=-1;;
337         }else{

```

```

338     _AppendFrontDigit (pResult, digit);
339     p1 = p1->prev;
340 }
341 }
342 if((p1->digit +flag)>0 && p1 == pA->pHead->next){
343     _AppendFrontDigit (pResult, p1->digit+flag);
344 }else{
345
346 }
347
348     return result;
349 }
350 struct UBigNumber MinusBN (struct UBigNumber *pA, struct UBigNumber *pB)//
有符号大数减
351 {
352     if(pA->pHead->digit==0 && pB->pHead->digit==0){
353         int flag=Compare(pA,pB);
354         if(flag==1 || flag==0){
355             return MinusUBN(pA,pB);
356         }else{
357             struct UBigNumber u = MinusUBN(pB,pA);
358             u.pHead->digit=1;
359             return u;
360         }
361     }else if(pA->pHead->digit==1 && pB->pHead->digit==1){
362         int flag=Compare(pA,pB);
363         if(flag==1 || flag==0){//前者大, 后者小, 相当于后减前, 加负号
364             return MinusUBN(pB,pA);
365         }else{
366             struct UBigNumber u = MinusUBN(pA,pB);//前减后, 加负号, 前小说明负
的多
367             u.pHead->digit=1;
368             return u;
369         }
370     }
371     }else if(pA->pHead->digit==1 && pB->pHead->digit==0){
372         struct UBigNumber u = AddUBN(pA,pB);
373         u.pHead->digit=1;
374         return u;
375     }else if(pA->pHead->digit==0 && pB->pHead->digit==1){
376         return AddUBN(pA,pB);
377     }
378
379 }
380 struct UBigNumber AddBN (struct UBigNumber *pA, struct UBigNumber *pB)//有
符号大数加
381 {
382     if(pA->pHead->digit==0 && pB->pHead->digit==0){
383         return AddUBN(pA,pB);
384     }else if(pA->pHead->digit==1 && pB->pHead->digit==1){
385         struct UBigNumber u = AddUBN(pA,pB);
386         u.pHead->digit=1;
387         return u;
388     }else if(pA->pHead->digit==0 && pB->pHead->digit==1){//b为负的
389         pB->pHead->digit=0;
390         int flag = Compare(pA,pB);
391         pB->pHead->digit=1;
392         if(flag==1 || flag ==0){

```

```

393         return MinusUBN(pA,pB);
394     }else if(flag == -1){
395         struct UBigNumber u = MinusUBN(pB,pA);
396         u.pHead->digit=1;
397         return u;
398     }
399
400     }else {//a为负的
401         pA->pHead->digit=0;
402         int flag = Compare(pA,pB);
403         pA->pHead->digit=1;
404         if(flag== -1 || flag ==0){
405             return MinusUBN(pB,pA);
406         }else if(flag == 1){
407             struct UBigNumber u = MinusUBN(pA,pB);
408             u.pHead->digit=1;
409             return u;
410         }
411     }
412 }
413 struct UBigNumber AddUBN (struct UBigNumber *pA, struct UBigNumber *pB)
414 {
415     struct UBigNumber result, *pResult = &result;
416     _InitUBN(pResult);
417     int iCarry = 0;
418     struct Node *p1, *p2;
419     p1 = pA->pTail;
420     p2 = pB->pTail;
421     while (p1 != pA->pHead && p2 != pB->pHead)
422     {
423         int digit = p1->digit + p2->digit + iCarry;
424         iCarry = digit / 10;
425         digit %= 10;
426         _AppendFrontDigit (pResult, digit);
427         p1 = p1->prev;
428         p2 = p2->prev;
429     }
430     while (p1 != pA->pHead)
431     {
432         int digit = p1->digit + iCarry;
433         iCarry = digit / 10;
434         digit %= 10;
435         _AppendFrontDigit (pResult, digit);
436         p1 = p1->prev;
437     }
438     while (p2 != pB->pHead)
439     {
440         int digit = p2->digit + iCarry;
441         iCarry = digit / 10;
442         digit %= 10;
443         _AppendFrontDigit (pResult, digit);
444         p2 = p2->prev;
445     }
446     if (iCarry != 0)
447         _AppendFrontDigit (pResult, iCarry);
448     return result;
449 }
450 void DestoryUBN (struct UBigNumber *pUBN)

```



```

451 {
452     while (pUBN->pHead != NULL)
453     {
454         struct Node *p = pUBN->pHead;
455         pUBN->pHead = p->next;
456         free (p);
457     }
458 }
459 void _InitUBN (struct UBigNumber *pUBN)
460 {
461     struct Node *p = _NewNode ();
462     p->digit=0;//代表是正的，后面输入符号会修改
463     pUBN->pHead = pUBN->pTail = p;
464     p->next = p->prev = NULL;
465     pUBN->digitCount = 0;
466 }
467 void _AppendDigit (struct UBigNumber *pUBN, int digit)
468 {
469     if (pUBN->digitCount == 1 && pUBN->pTail->digit == 0)
470     { //直到出现非0数字才可以结束
471         pUBN->pTail->digit = digit;
472         return;
473     }
474     struct Node *p = _NewNode (); //数字链表添加一个结点
475     p->digit = digit;
476     p->next = NULL;
477     p->prev = pUBN->pTail;
478     pUBN->pTail->next = p;
479     pUBN->pTail = p;
480     ++pUBN->digitCount;
481 }
482 void _AppendFrontDigit (struct UBigNumber *pUBN, int digit)
483 {
484     struct Node *p = _NewNode ();
485     p->digit = digit;
486     p->next = pUBN->pHead->next;
487     if (p->next != NULL)
488         p->next->prev = p;
489     p->prev = pUBN->pHead;
490     pUBN->pHead->next = p;
491     if (pUBN->pTail == pUBN->pHead)
492         pUBN->pTail = p;
493     ++pUBN->digitCount;
494 }
495 void _Normalize (struct UBigNumber *pUBN)
496 {
497     if (pUBN->digitCount == 0)
498         _AppendDigit (pUBN, 0);
499     while (pUBN->digitCount > 1 && pUBN->pHead->next->digit == 0)
500     { //过滤0
501         struct Node *p;
502         p = pUBN->pHead->next;
503         pUBN->pHead->next = p->next;
504         p->next->prev = pUBN->pHead;
505         free (p);
506         --pUBN->digitCount;
507     }
508 }

```

```

509
510 struct Node *_NewNode ()
511 {
512     struct Node *p;
513     p = (struct Node *) malloc (sizeof (struct Node));
514     if (p == NULL)
515     {
516         printf ("Error : out of memory\n");
517         exit (-1);
518     }
519     return p;
520 }

```

#### 测试用例

```

1  -1234567890987654321333888999666
2  -147655765659657669789687967867

```

重置测试用例

 运行测试

运行结束 测试于 6秒前 C (gcc 6.5.0)

运行结果

编译器输出

运行结果

预期结果

```

1  -1086912125327996651544201031799
2  -1382223656647311991123576967533
3  -18229106720261088253819676708754688731363

```

```

1  -1086912125327996651544201031799
2  -1382223656647311991123576967533
3  -18229106720261088253819676708754688731363

```