

```

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
30
31
32 int main ()
33 {
34     struct UBigNumber A, B, C,D;
35     A = InputUBN ();
36     B = InputUBN ();
37     /*    C = AddUBN (&A, &B);
38         D = MinusUBN(&A, &B);
39
40         PrintUBN (A);
41         printf (" + ");
42         PrintUBN (B);
43         printf (" = ");
44         PrintUBN (C);
45         printf("\n");
46         PrintUBN (A);
47         printf (" - ");
48         PrintUBN (B);
49         printf (" = ");
50         PrintUBN (D);    */
51     int com = Compare(&A,&B);
52     if(com==1){
        printf("A>B");
    }

```

```

53     }else if(com==-1){
54         printf("A<B");
55     }else{
56         printf("A=B");
57     }
58     printf("有符号大数加减: ");
59     PrintUBN (A);
60     printf (" + ");
61     PrintUBN (B);
62     printf (" = ");
63     PrintUBN (AddBN(&A,&B));
64     printf("\n");
65     PrintUBN (A);
66     printf (" - ");
67     PrintUBN (B);
68     printf (" = ");
69     PrintUBN (MinusBN(&A,&B));
70     DestoryUBN (&A);
71     DestoryUBN (&B);
72     //DestoryUBN (&C);
73     return 0;
74 }
75 int Compare(struct UBigNumber *pA, struct UBigNumber *pB){
76     if(pA->pHead->digit==1 && pB->pHead->digit==0){//前 负, 后 正
77         return -1;//后面大
78     }else if(pA->pHead->digit==0 && pB->pHead->digit==1){
79         return 1;//前面大
80     }else if(pA->pHead->digit==0 && pB->pHead->digit==0){
81         //先比较位数, 多者 大
82         int a=0,b=0;
83         struct Node *p1,*p2;
84         p1=pA->pHead->next;
85         p2=pB->pHead->next;
86         while(p1!=pA->pTail){
87             a++;
88             p1=p1->next;
89         }
90         a++;
91         while(p2!=pB->pTail){
92             b++;
93             p2=p2->next;
94         }
95         b++;
96         if(a>b){
97             return 1;
98         }else if(a<b){
99             return -1;
100        }else{//位数相等时, 从头开始比, 先大者大
101            p1=pA->pHead->next;
102            p2=pB->pHead->next;
103            while(p1!=pA->pTail && p2!=pB->pTail){
104                if(p1->digit > p2->digit){
105                    return 1;
106                }else if(p1->digit < p2->digit){
107                    return -1;
108                }else {
109                    p1=p1->next;
110                    p2=p2->next;

```

```

111         }
112     }
113 }
114     return 0;//两者相等
115 }
116 }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
117     //先比较位数，多者 小
118     int a=0,b=0;
119     struct Node *p1,*p2;
120     p1=pA->pHead->next;
121     p2=pB->pHead->next;
122     while(p1!=pA->pTail){
123         a++;
124         p1=p1->next;
125     }
126     a++;
127     while(p2!=pB->pTail){
128         b++;
129         p2=p2->next;
130     }
131     b++;
132     if(a>b){
133         return -1;
134     }else if(a<b){
135         return 1;
136     }else{//位数相等时，从头开始比，先大者小
137         p1=pA->pHead->next;
138         p2=pB->pHead->next;
139         while(p1!=pA->pTail && p2!=pB->pTail){
140             if(p1->digit > p2->digit){
141                 return -1;
142             }else if(p1->digit < p2->digit){
143                 return 1;
144             }else {
145                 p1=p1->next;
146                 p2=p2->next;
147             }
148         }
149     }
150     return 0;//两者相等
151 }
152 }
153 }
154
155 struct UBigNumber InputUBN ()
156 {
157     struct UBigNumber result;
158     _InitUBN(&result);
159
160     char ch;
161     do
162         ch = getchar ();
163     while ((ch < '0' || ch > '9') && ch!='-');
164     if(ch=='-'){
165         result.pHead->digit=1;//1代表是负数
166         ch = getchar ();
167     }
168     while (ch >= '0' && ch <= '9')

```

```

169     {
170         _AppendDigit (&result, ch - '0');
171         ch = getchar ();
172     }
173     _Normalize(&result);
174     return result;
175 }
176 void PrintUBN (struct UBigNumber ubn)
177 {
178     assert (ubn.digitCount > 0 && ubn.pHead->next != NULL);
179     struct Node *la = ubn.pHead->next;
180     if(ubn.pHead->digit==1){
181         printf("-");
182     }
183     while (la)
184     {
185         printf ("%d", la->digit);
186         la = la->next;
187     }
188 }
189 struct UBigNumber MinusUBN (struct UBigNumber *pA, struct UBigNumber *pB)
190 {
191     struct UBigNumber result, *pResult = &result;
192     int flag=0;
193     _InitUBN(pResult);
194     struct Node *p1, *p2;
195     p1 = pA->pTail;
196     p2 = pB->pTail;
197     while (p1 != pA->pHead && p2 != pB->pHead)
198     {
199         int digit = p1->digit - p2->digit +flag;
200         flag=0;
201         if(digit>=0){
202             _AppendFrontDigit (pResult, digit);
203             p1 = p1->prev;
204             p2 = p2->prev;
205         }else{
206             _AppendFrontDigit (pResult, digit+10);
207             p1 = p1->prev;
208             p2 = p2->prev;
209             flag=-1;
210         }
211     }
212     while (p1 != pA->pHead->next && p1 != pA->pHead)
213     {
214         int digit = (p1->digit) + flag;
215         flag=0;
216         if(digit<0){
217             _AppendFrontDigit (pResult, digit+10);
218             p1 = p1->prev;
219             flag=-1;;
220         }else{
221             _AppendFrontDigit (pResult, digit);
222             p1 = p1->prev;
223         }
224     }
225     if((p1->digit +flag)>0 && p1 == pA->pHead->next){
226         _AppendFrontDigit (pResult, p1->digit+flag);

```

```

227     }else{
228
229     }
230
231     return result;
232 }
233 struct UBigNumber MinusBN (struct UBigNumber *pA, struct UBigNumber *pB)//
有符号大数减
234 {
235     if(pA->pHead->digit==0 && pB->pHead->digit==0){
236         int flag=Compare(pA,pB);
237         if(flag==1 || flag==0){
238             return MinusUBN(pA,pB);
239         }else{
240             struct UBigNumber u = MinusUBN(pB,pA);
241             u.pHead->digit=1;
242             return u;
243         }
244     }else if(pA->pHead->digit==1 && pB->pHead->digit==1){
245         int flag=Compare(pA,pB);
246         if(flag==1 || flag==0){//前者大，后者小，相当于后减前，加负号
247             return MinusUBN(pB,pA);
248         }else{
249             struct UBigNumber u = MinusUBN(pA,pB);//前减后，加负号，前小说明负
的多
250             u.pHead->digit=1;
251             return u;
252         }
253     }
254     }else if(pA->pHead->digit==1 && pB->pHead->digit==0){
255         struct UBigNumber u = AddUBN(pA,pB);
256         u.pHead->digit=1;
257         return u;
258     }else if(pA->pHead->digit==0 && pB->pHead->digit==1){
259         return AddUBN(pA,pB);
260     }
261
262 }
263 struct UBigNumber AddBN (struct UBigNumber *pA, struct UBigNumber *pB)//有
符号大数加
264 {
265     if(pA->pHead->digit==0 && pB->pHead->digit==0){
266         return AddUBN(pA,pB);
267     }else if(pA->pHead->digit==1 && pB->pHead->digit==1){
268         struct UBigNumber u = AddUBN(pA,pB);
269         u.pHead->digit=1;
270         return u;
271     }else if(pA->pHead->digit==0 && pB->pHead->digit==1){//b为负的
272         pB->pHead->digit=0;
273         int flag = Compare(pA,pB);
274         pB->pHead->digit=1;
275         if(flag==1 || flag ==0){
276             return MinusUBN(pA,pB);
277         }else if(flag == -1){
278             struct UBigNumber u = MinusUBN(pB,pA);
279             u.pHead->digit=1;
280             return u;
281         }

```

```

282
283     }else { //a为负的
284         pA->pHead->digit=0;
285         int flag = Compare(pA,pB);
286         pA->pHead->digit=1;
287         if(flag== -1 || flag == 0){
288             return MinusUBN(pB,pA);
289         }else if(flag == 1){
290             struct UBigNumber u = MinusUBN(pA,pB);
291             u.pHead->digit=1;
292             return u;
293         }
294     }
295 }
296 struct UBigNumber AddUBN (struct UBigNumber *pA, struct UBigNumber *pB)
297 {
298     struct UBigNumber result, *pResult = &result;
299     _InitUBN(pResult);
300     int iCarry = 0;
301     struct Node *p1, *p2;
302     p1 = pA->pTail;
303     p2 = pB->pTail;
304     while (p1 != pA->pHead && p2 != pB->pHead)
305     {
306         int digit = p1->digit + p2->digit + iCarry;
307         iCarry = digit / 10;
308         digit %= 10;
309         _AppendFrontDigit (pResult, digit);
310         p1 = p1->prev;
311         p2 = p2->prev;
312     }
313     while (p1 != pA->pHead)
314     {
315         int digit = p1->digit + iCarry;
316         iCarry = digit / 10;
317         digit %= 10;
318         _AppendFrontDigit (pResult, digit);
319         p1 = p1->prev;
320     }
321     while (p2 != pB->pHead)
322     {
323         int digit = p2->digit + iCarry;
324         iCarry = digit / 10;
325         digit %= 10;
326         _AppendFrontDigit (pResult, digit);
327         p2 = p2->prev;
328     }
329     if (iCarry != 0)
330         _AppendFrontDigit (pResult, iCarry);
331     return result;
332 }
333 void DestoryUBN (struct UBigNumber *pUBN)
334 {
335     while (pUBN->pHead != NULL)
336     {
337         struct Node *p = pUBN->pHead;
338         pUBN->pHead = p->next;
339         free (p);

```

```

340     }
341 }
342 void _InitUBN (struct UBigNumber *pUBN)
343 {
344     struct Node *p = _NewNode ();
345     p->digit=0;//代表是正的，后面输入符号会修改
346     pUBN->pHead = pUBN->pTail = p;
347     p->next = p->prev = NULL;
348     pUBN->digitCount = 0;
349 }
350 void _AppendDigit (struct UBigNumber *pUBN, int digit)
351 {
352     if (pUBN->digitCount == 1 && pUBN->pTail->digit == 0)
353     { //直到出现非0数字才可以结束
354         pUBN->pTail->digit = digit;
355         return;
356     }
357     struct Node *p = _NewNode (); //数字链表添加一个结点
358     p->digit = digit;
359     p->next = NULL;
360     p->prev = pUBN->pTail;
361     pUBN->pTail->next = p;
362     pUBN->pTail = p;
363     ++pUBN->digitCount;
364 }
365 void _AppendFrontDigit (struct UBigNumber *pUBN, int digit)
366 {
367     struct Node *p = _NewNode ();
368     p->digit = digit;
369     p->next = pUBN->pHead->next;
370     if (p->next != NULL)
371         p->next->prev = p;
372     p->prev = pUBN->pHead;
373     pUBN->pHead->next = p;
374     if (pUBN->pTail == pUBN->pHead)
375         pUBN->pTail = p;
376     ++pUBN->digitCount;
377 }
378 void _Normalize (struct UBigNumber *pUBN)
379 {
380     if (pUBN->digitCount == 0)
381         _AppendDigit (pUBN, 0);
382     while (pUBN->digitCount > 1 && pUBN->pHead->next->digit == 0)
383     { //过滤0
384         struct Node *p;
385         p = pUBN->pHead->next;
386         pUBN->pHead->next = p->next;
387         p->next->prev = pUBN->pHead;
388         free (p);
389         --pUBN->digitCount;
390     }
391 }
392
393 struct Node *_NewNode ()
394 {
395     struct Node *p;
396     p = (struct Node *) malloc (sizeof (struct Node));
397     if (p == NULL)

```

```

398     {
399         printf ("Error : out of memory\n");
400         exit (-1);
401     }
402     return p;
403 }

```

```

6912125327996651544201031799
PS D:\csjjg\程序设计综合实践> cd "d:\csjjg\程序设计综合实践\" ; if ($?) { gcc ninth.c -o ninth } ; if ($?) { .\ninth }
-1234567890987654321333888999666
147655765659657669789687967867
A<B有符号大数加减: -1234567890987654321333888999666 + 1476557656596576
69789687967867 = -1086912125327996651544201031799
-1234567890987654321333888999666 - 147655765659657669789687967867 = -1
382223656647311991123576967533

```