

```

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 MyMinus(struct UBigNumber *pA, struct UBigNumber *pB);
26
27 int main ()
28 {
29     struct UBigNumber A, B, C,D;
30     A = InputUBN ();
31     B = InputUBN ();
32     C = AddUBN (&A, &B);
33     D = MyMinus(&A, &B);
34
35     PrintUBN (A);
36     printf (" + ");
37     PrintUBN (B);
38     printf (" = ");
39     PrintUBN (C);
40     printf("\n");
41     PrintUBN (A);
42     printf (" - ");
43     PrintUBN (B);
44     printf (" = ");
45     PrintUBN (D);
46
47     DestoryUBN (&A);
48     DestoryUBN (&B);
49     DestoryUBN (&C);
50     return 0;
51 }
52
53 struct UBigNumber InputUBN ()
54 {
55     struct UBigNumber result;

```

```

56     _InitUBN(&result);
57
58     char ch;
59     do
60         ch = getchar ();
61     while (ch < '0' || ch > '9');
62     while (ch >= '0' && ch <= '9')
63     {
64         _AppendDigit (&result, ch - '0');
65         ch = getchar ();
66     }
67     _Normalize(&result);
68     return result;
69 }
70 void PrintUBN (struct UBigNumber ubn)
71 {
72     assert (ubn.digitCount > 0 && ubn.pHead->next != NULL);
73     struct Node *la = ubn.pHead->next;
74     while (la)
75     {
76         printf ("%d", la->digit);
77         la = la->next;
78     }
79 }
80 struct UBigNumber MyMinus (struct UBigNumber *pA, struct UBigNumber *pB)
81 {
82     struct UBigNumber result, *pResult = &result;
83     int flag=0;
84     _InitUBN(pResult);
85     struct Node *p1, *p2;
86     p1 = pA->pTail;
87     p2 = pB->pTail;
88     while (p1 != pA->pHead && p2 != pB->pHead)
89     {
90         int digit = p1->digit - p2->digit +flag;
91         flag=0;
92         if(digit>=0){
93             _AppendFrontDigit (pResult, digit);
94             p1 = p1->prev;
95             p2 = p2->prev;
96         }else{
97             _AppendFrontDigit (pResult, digit+10);
98             p1 = p1->prev;
99             p2 = p2->prev;
100             flag=-1;
101         }
102     }
103     while (p1 != pA->pHead->next && p1 != pA->pHead)
104     {
105         int digit = (p1->digit) + flag;
106         flag=0;
107         if(digit<0){
108             _AppendFrontDigit (pResult, digit+10);
109             p1 = p1->prev;
110             flag=-1;;
111         }else{
112             _AppendFrontDigit (pResult, digit);
113             p1 = p1->prev;

```

```

114     }
115 }
116 if((p1->digit +flag)>0 && p1 == pA->pHead->next){
117     _AppendFrontDigit (pResult, p1->digit+flag);
118 }else{
119
120 }
121
122     return result;
123 }
124 struct UBigNumber AddUBN (struct UBigNumber *pA, struct UBigNumber *pB)
125 {
126     struct UBigNumber result, *pResult = &result;
127     _InitUBN(pResult);
128     int iCarry = 0;
129     struct Node *p1, *p2;
130     p1 = pA->pTail;
131     p2 = pB->pTail;
132     while (p1 != pA->pHead && p2 != pB->pHead)
133     {
134         int digit = p1->digit + p2->digit + iCarry;
135         iCarry = digit / 10;
136         digit %= 10;
137         _AppendFrontDigit (pResult, digit);
138         p1 = p1->prev;
139         p2 = p2->prev;
140     }
141     while (p1 != pA->pHead)
142     {
143         int digit = p1->digit + iCarry;
144         iCarry = digit / 10;
145         digit %= 10;
146         _AppendFrontDigit (pResult, digit);
147         p1 = p1->prev;
148     }
149     while (p2 != pB->pHead)
150     {
151         int digit = p2->digit + iCarry;
152         iCarry = digit / 10;
153         digit %= 10;
154         _AppendFrontDigit (pResult, digit);
155         p2 = p2->prev;
156     }
157     if (iCarry != 0)
158         _AppendFrontDigit (pResult, iCarry);
159     return result;
160 }
161 void DestoryUBN (struct UBigNumber *pUBN)
162 {
163     while (pUBN->pHead != NULL)
164     {
165         struct Node *p = pUBN->pHead;
166         pUBN->pHead = p->next;
167         free (p);
168     }
169 }
170 void _InitUBN (struct UBigNumber *pUBN)
171 {

```

```

172     struct Node *p = _NewNode ();
173     PUBN->pHead = PUBN->pTail = p;
174     p->next = p->prev = NULL;
175     PUBN->digitCount = 0;
176 }
177 void _AppendDigit (struct UBigNumber *PUBN, int digit)
178 {
179     if (PUBN->digitCount == 1 && PUBN->pTail->digit == 0)
180     { //直到出现非0数字才可以结束
181         PUBN->pTail->digit = digit;
182         return;
183     }
184     struct Node *p = _NewNode (); //数字链表添加一个结点
185     p->digit = digit;
186     p->next = NULL;
187     p->prev = PUBN->pTail;
188     PUBN->pTail->next = p;
189     PUBN->pTail = p;
190     ++PUBN->digitCount;
191 }
192 void _AppendFrontDigit (struct UBigNumber *PUBN, int digit)
193 {
194     struct Node *p = _NewNode ();
195     p->digit = digit;
196     p->next = PUBN->pHead->next;
197     if (p->next != NULL)
198         p->next->prev = p;
199     p->prev = PUBN->pHead;
200     PUBN->pHead->next = p;
201     if (PUBN->pTail == PUBN->pHead)
202         PUBN->pTail = p;
203     ++PUBN->digitCount;
204 }
205 void _Normalize (struct UBigNumber *PUBN)
206 {
207     if (PUBN->digitCount == 0)
208         _AppendDigit (PUBN, 0);
209     while (PUBN->digitCount > 1 && PUBN->pHead->next->digit == 0)
210     { //过滤0
211         struct Node *p;
212         p = PUBN->pHead->next;
213         PUBN->pHead->next = p->next;
214         p->next->prev = PUBN->pHead;
215         free (p);
216         --PUBN->digitCount;
217     }
218 }
219
220 struct Node *_NewNode ()
221 {
222     struct Node *p;
223     p = (struct Node *) malloc (sizeof (struct Node));
224     if (p == NULL)
225     {
226         printf ("Error : out of memory\n");
227         exit (-1);
228     }
229     return p;

```

```
230 }  
231
```

```
4 6 5 2 9 1 3 7 8 10  
PS D:\csjjg\程序设计综合实践> cd "d:\csjjg\程序设计综合实践\" ; if ($?  
) { gcc eighth.c -o eighth } ; if ($?) { .\eighth }  
1234567890987654321333888999666  
147655765659657669789687967867  
1234567890987654321333888999666 + 147655765659657669789687967867 = 138  
2223656647311991123576967533  
1234567890987654321333888999666 - 147655765659657669789687967867 = 108  
6912125327996651544201031799  
PS D:\csjjg\程序设计综合实践>
```