

```

1  #define _CRTDBG_MAP_ALLOC
2  #include <stdlib.h>
3  #include <crtDBG.h>
4
5  #ifdef _DEBUG
6  #ifndef DBG_NEW
7  #define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
8  #define new DBG_NEW
9  #endif
10 #endif // _DEBUG
11 //-----
12 #include <iostream>
13 using namespace std;
14 #include "BigInt.h"
15 #include <string.h>
16 //-----
17
18 //The n'th Fibonacci number n(1)=1,n(2)=1,n(3)=2,n(4)=3,n(5)=5,...
19 BigInt fibo(unsigned int n)
20 {
21     //in case i dont need to calculate the organ of fibonacci series:
22     BigInt n_fib_number = BigInt_create(0);
23
24     if (n == 1 || n == 2)
25     {
26         BigInt_inc(n_fib_number);
27     }
28     else if (n > 2) // in case i need to calculate the organ of fibonacci ↗
29         series , must initialize fibonacci series and then calculate:
30     {
31         //initialize fibonacci series:
32         BigInt a = BigInt_create(1);
33         BigInt b = BigInt_create(1);
34         BigInt tmp;
35         //calculating The n'th Fibonacci number
36         for (unsigned int i = 2; i < n; i++)
37         {
38             tmp = a + b;
39             BigInt_assign(n_fib_number, tmp);
40             BigInt_assign(a, b);
41             BigInt_assign(b, n_fib_number);
42             BigInt_destroy(tmp);
43         }
44         BigInt_destroy(a);
45         BigInt_destroy(b);
46     }
47     return n_fib_number;
48 }
49
50 //sqrt(x)
51 BigInt sqrt(const BigInt& x)
52 {

```

```
53     BigInt sum = BigInt_create(0);
54     BigInt odd = BigInt_create(1);
55     BigInt sq = BigInt_create(0);
56     BigInt const2 = BigInt_create(2);
57     BigInt tmp;
58     while (sum < x)
59     {
60         tmp = sum + odd;
61         BigInt_assign(sum, tmp);
62         BigInt_destroy(tmp);
63
64         BigInt_inc(sq);
65
66         tmp = odd + const2;
67         BigInt_assign(odd, tmp);
68         BigInt_destroy(tmp);
69     }
70     BigInt_destroy(sum);
71     BigInt_destroy(odd);
72     BigInt_destroy(const2);
73
74     return sq;
75 }
76
77 ////////////////////////////////////////////////////
78 //operator overload:
79
80 int operator<(const BigInt& a, const BigInt& b)
81 {
82     int ans = BigInt_compare(a, b);
83     if (ans == -1)
84         ans = 1;
85     else
86         ans = 0;
87
88     return ans;
89 }
90
91 int operator==(const BigInt& a, const BigInt& b)
92 {
93     if (BigInt_compare(a, b) == 0)
94         return 1;
95     else
96         return 0;
97 }
98
99 BigInt operator+(const BigInt& a, const BigInt& b)
100 {
101     return BigInt_add(a, b);
102 }
103
104
105
```