

大程 10report

运用 VS2017professional 和 c 语言代码如下：

代码如下，分别实现了浮点数到字符串，字符串到浮点数。浮点数的加减乘除。

```
1. #include<stdio.h>
2. #include<stdlib.h>
3.
4. typedef unsigned int dwrd; //32 - bit
5. char* ftoa(dwrd num)
6. {
7.     int i = 0;
8.     char *str = (char *)malloc(sizeof(char) * 32);
9.     for (int i = 0; i < 32; i++)
10.    {
11.        if (num&(0x00000001 << i))str[i] = '1';
12.        else str[i] = '0';
13.    }
14.    return str;
15. }
16.
17. dwrd atof(char* str)
18. {
19.     dwrd a=0x00000000;
20.     for (int i = 0; i < 32; i++)
21.     {
22.         if (str[i] == '1')
23.             a | (0x00000001 << i);
24.
25.     }
26.     return a;
27. }
28.
29. dwrd fadd(dwrd num1, dwrd num2)
30. {
31.     int temp1, temp2;
32.     int e;
33.     if (num1 & 0x80000000)
34.     {
35.         if (num2 & 0x80000000)
36.         {
37.             return -fadd(num1 & 0x7FFFFFFF, num2 & 0x7FFFFFFF);
38.         }
```

```

39.         else
40.         {
41.             return fsub(num2 & 0x7FFFFFFF, num1 & 0x7FFFFFFF);
42.         }
43.     }
44.     else
45.     {
46.         if (num2 & 0x80000000)
47.         {
48.             return fsub(num1 & 0x7FFFFFFF, num2 & 0x7FFFFFFF);
49.         }
50.         else
51.         {
52.             temp1 = num1 >> 23;
53.             temp2 = num2 >> 23;
54.             if (temp1 > temp2)
55.             {
56.                 e = temp1 - temp2;
57.                 temp1 = num1 & 0x007FFFFF;
58.                 temp2 = num2 & 0x007FFFFF;
59.                 temp2 = temp2 >> e;
60.                 temp1 = temp1 + temp2;
61.                 temp1 = temp1 | (num1 & 0x7F800000);
62.                 return temp1;
63.             }
64.             else
65.             {
66.                 e = temp2 - temp1;
67.                 temp2 = num1 & 0x007FFFFF;
68.                 temp1 = num2 & 0x007FFFFF;
69.                 temp1 = temp1 >> e;
70.                 temp2 = temp2 + temp1;
71.                 temp2 = temp2 | (num2 & 0x7F800000);
72.                 return temp2;
73.             }
74.         }
75.     }
76. }
77.
78. dword fsub(dword num1, dword num2)
79. {
80.     int temp1, temp2;
81.     int e;
82.     if (num1 & 0x80000000)

```

```

83.     {
84.         if (num2 & 0x80000000)
85.         {
86.             return fsub(num2 & 0x7FFFFFFF, num1 & 0x7FFFFFFF);
87.         }
88.         else
89.         {
90.             return -fadd(num2 & 0x7FFFFFFF, num1 & 0x7FFFFFFF);
91.         }
92.     }
93.     else
94.     {
95.         if (num2 & 0x80000000)
96.         {
97.             return fadd(num1 & 0x7FFFFFFF, num2 & 0x7FFFFFFF);
98.         }
99.         else
100.        {
101.            temp1 = num1 >> 23;
102.            temp2 = num2 >> 23;
103.            if (temp1 > temp2)
104.            {
105.                e = temp1 - temp2;
106.                temp1 = num1 & 0x007FFFFF;
107.                temp2 = num2 & 0x007FFFFF;
108.                temp2 = temp2 >> e;
109.                temp1 = temp1 - temp2;
110.                temp1 = temp1 | (num1 & 0x7F800000);
111.                return temp1;
112.            }
113.            else
114.            {
115.                e = temp2 - temp1;
116.                temp2 = num1 & 0x007FFFFF;
117.                temp1 = num2 & 0x007FFFFF;
118.                temp1 = temp1 >> e;
119.                temp2 = temp2 - temp1;
120.                temp2 = temp2 | (num2 & 0x7F800000);
121.                return temp2;
122.            }
123.        }
124.    }
125. }
126.

```

```

127. dwrdr fmul(dwrdr num1, dwrdr num2)
128. {
129.     int temp1, temp2;
130.     int e;
131.     if (num1 & 0x80000000)
132.     {
133.         if (num2 & 0x80000000)
134.         {
135.             return fmul(num2 & 0x7FFFFFFF, num1 & 0x7FFFFFFF);
136.         }
137.         else
138.         {
139.             return -fmul(num2 & 0x7FFFFFFF, num1 & 0x7FFFFFFF);
140.         }
141.     }
142.     else
143.     {
144.         if (num2 & 0x80000000)
145.         {
146.             return -fmul(num1 & 0x7FFFFFFF, num2 & 0x7FFFFFFF);
147.         }
148.         else
149.         {
150.             temp1 = num1 >> 23;
151.             temp2 = num2 >> 23;
152.             e = temp1 + temp2;
153.             temp1 = num1 & 0x007FFFFFFF;
154.             temp2 = num2 & 0x007FFFFFFF;
155.             temp1 = temp1 * temp2;
156.             temp1 = temp1 | (e << 23);
157.             return temp1;
158.         }
159.     }
160. }
161.
162. dwrdr fdiv(dwrdr num1, dwrdr num2)
163. {
164.     int temp1, temp2;
165.     int e;
166.     if (num1 & 0x80000000)
167.     {
168.         if (num2 & 0x80000000)
169.         {
170.             return fdiv(num2 & 0x7FFFFFFF, num1 & 0x7FFFFFFF);

```

```
171.     }
172.     else
173.     {
174.         return -fdiv(num2 & 0x7FFFFFFF, num1 & 0x7FFFFFFF);
175.     }
176. }
177. else
178. {
179.     if (num2 & 0x80000000)
180.     {
181.         return -fdiv(num1 & 0x7FFFFFFF, num2 & 0x7FFFFFFF);
182.     }
183.     else
184.     {
185.         temp1 = num1 >> 23;
186.         temp2 = num2 >> 23;
187.         if (temp1 > temp2)
188.         {
189.             e = temp1 - temp2;
190.             temp1 = num1 & 0x007FFFFFFF;
191.             temp2 = num2 & 0x007FFFFFFF;
192.             temp2 = temp2 >> e;
193.             temp1 = temp1 / temp2;
194.             temp1 = temp1 | (e << 23);
195.             return temp1;
196.         }
197.         else
198.         {
199.             e = temp2 - temp1;
200.             temp2 = num1 & 0x007FFFFFFF;
201.             temp1 = num2 & 0x007FFFFFFF;
202.             temp1 = temp1 >> e;
203.             temp2 = temp2 / temp1;
204.             temp2 = temp2 | (e << 23);
205.             return temp2;
206.         }
207.     }
208. }
209. }
```