

C++知识点总结(4): 高精度加减法

原创

AICodeThunder

已于 2023-11-30 18:55:56 修改

阅读量261

收藏 3

点赞数

分类专栏:


C++知识点总结

文章标签:

C/C++

编程笔记

知识点总结

 C++知识点总结 专栏收录该内容

7 订阅 53 篇文章

一、什么是高精度数

- 【含义】无法使用基础类型存储的数字。
- 【存储】这里使用 `char` 数组，更稳定一些。

二、输入

题目描述

输入两个高精度数（不超过 10^{500} ），并输出两个高精度数。

提示

数组的长度直接用505即可，别看 10_{500} 很大，其实只有501位。

参考答案

```
1  #include <iostream>
2  #include <cstring>
3  using namespace std;
4
5  int main()
6  {
7      char num1[505] = {};
8      char num2[505] = {};
9      cin >> num1 >> num2;
10     cout << num1 << endl << num2;
11     return 0;
12 }
```

三、转换为整型&倒序存储

```
1  // 存储的整型数组
2  int num1_int[505] = {};
3  int num2_int[505] = {};
4
5  // 取两个数字的长度
6  int len_num1 = strlen(num1);
7  int len_num2 = strlen(num2);
8
9  // 遍历存储
10 for (int i = 0; i <= len_num1-1; i++)
11     num1_int[len_num1-i-1] = num1[i] - 48;
12 for (int i = 0; i <= len_num2-1; i++)
13     num2_int[len_num2-i-1] = num2[i] - 48;
```

四、计算

 AICodeThunder

关注

👍 0

🗨 0

🌟 3

💬 0

1. 得到加的次数

```
1 | int len_ans = max(len_num1, len_num2);
```

2. 遍历做加法

(1) 没有进位时

```
1 | #include <iostream>
2 | #include <cstring>
3 | using namespace std;
4 |
5 | int main()
6 | {
7 |     char num1[505] = {};
8 |     char num2[505] = {};
9 |     cin >> num1 >> num2;
10 |
11 |
12 |     // 存储的整型数组
13 |     int num1_int[505] = {};
14 |     int num2_int[505] = {};
15 |
16 |     // 取两个数字的长度
17 |     int len_num1 = strlen(num1);
18 |     int len_num2 = strlen(num2);
19 |
20 |     // 遍历存储
21 |     for (int i = 0; i <= len_num1-1; i++)
22 |     {
23 |         num1_int[len_num1-i-1] = num1[i] - 48;
24 |     }
25 |     for (int i = 0; i <= len_num2-1; i++)
26 |     {
27 |         num2_int[len_num2-i-1] = num2[i] - 48;
28 |     }
29 |
30 |
31 |     // 存储运算结果
32 |     int len_ans = max(len_num1, len_num2);
33 |     int ans[505] = {};
34 |     for (int i = 0; i <= len_ans-1; i++)
35 |     {
36 |         ans[i] = num1_int[i] + num2_int[i];
37 |     }
38 |
39 |
40 |     // 倒序输出运算结果
41 |     for (int i = len_ans-1; i >= 0; i--)
42 |     {
43 |         cout << ans[i];
44 |     }
45 |     return 0;
46 | }
```

(2) 有进位时完善后的代码

```
1 | #include <iostream>
2 | #include <cstring>
3 | using namespace std;
4 |
5 | int main()
6 | {
7 |     char num1[505] = {};
8 |     char num2[505] = {};
9 |     cin >> num1 >> num2;
10 |
11 |
12 |     // 存储的整型数组
13 |     int num1_int[505] = {};
```



AI Code Thunder

关注

👍 0



🌟 3

💬 0

```
15     int num2_int[505] = {};  
16  
17     // 取两个数字的长度  
18     int len_num1 = strlen(num1);  
19     int len_num2 = strlen(num2);  
20  
21     // 遍历存储  
22     for (int i = 0; i <= len_num1-1; i++)  
23     {  
24         num1_int[len_num1-i-1] = num1[i] - 48;  
25     }  
26     for (int i = 0; i <= len_num2-1; i++)  
27     {  
28         num2_int[len_num2-i-1] = num2[i] - 48;  
29     }  
30  
31     // 计算  
32     int len_ans = max(len_num1, len_num2);  
33     int ans[505] = {};  
34     int in = 0;  
35     for (int i = 0; i <= len_ans-1; i++)  
36     {  
37         ans[i] = num1_int[i] + num2_int[i] + in;  
38         in = ans[i] / 10; // 进位的结果  
39         ans[i] %= 10; // 存储真实结果的最后一位  
40     }  
41  
42     // 特例先行  
43     if (in > 0)  
44     {  
45         ans[len_ans] = in;  
46         len_ans++;  
47     }  
48  
49     for (int i = len_ans-1; i >= 0; i--)  
50     {  
51         cout << ans[i];  
52     }  
53     return 0;  
}
```

文A

附录

高精度减法

```
1  int ans[505] = {};  
2  int len_ans = len_num1; // 默认大数-小数  
3  for (int i = 0; i <= len_ans-1; i++)  
4  {  
5      if (num1_int[i] < num2_int[i]) // 借位的情况  
6      {  
7          num1_int[i+1]--; // 借一  
8          num1_int[i] += 10; // 加10  
9      }  
10     ans[i] = num1_int[i] - num2_int[i]; // 减法  
11 }  
12  
13 // 输出前写下面的代码  
14 while(ans[len_ans-1] == 0 && len_ans > 1) // 最高位的情况, 前导0且结果非0的情况  
15 {  
16     len_ans--; // 减少, 后面不会输出前面的0  
17 }
```

完整代码

```
1 #include <iostream>  
2 #include <cstring>  
3 using namespace std;  
4  
-
```



AI Code Thunder

关注

👍 0

🗨 0

🌟 3

💬 0

```
5 int main()
6 {
7     char num1[505] = {};
8     char num2[505] = {};
9     cin >> num1 >> num2;
10
11
12     // 存储的整型数组
13     int num1_int[505] = {};
14     int num2_int[505] = {};
15
16     // 取两个数字的长度
17     int len_num1 = strlen(num1);
18     int len_num2 = strlen(num2);
19
20     // 遍历存储
21     for (int i = 0; i <= len_num1-1; i++)
22     {
23         num1_int[len_num1-i-1] = num1[i] - 48;
24     }
25     for (int i = 0; i <= len_num2-1; i++)
26     {
27         num2_int[len_num2-i-1] = num2[i] - 48;
28     }
29
30     // 计算
31     int ans[505] = {};
32     int len_ans = len_num1; // 默认大数-小数
33     for (int i = 0; i <= len_ans-1; i++)
34     {
35         if (num1_int[i] < num2_int[i]) // 借位的情况
36         {
37             num1_int[i+1]--; // 借一
38             num1_int[i] += 10; // 加10
39         }
40         ans[i] = num1_int[i] - num2_int[i]; // 减法
41     }
42
43     // 输出前写下面的代码
44     while(ans[len_ans-1] == 0 && len_ans > 1) // 特例先行
45     {
46         len_ans--; // 不会输出前面的0了
47     }
48
49     for (int i = len_ans-1; i >= 0; i--)
50     {
51         cout << ans[i];
52     }
53     return 0;
54 }
```



需要所有完整代码，进入：

完整代码整理

[完整代码总结获取入口](#)

文章知识点与官方知识档案匹配，可进一步学习相关知识

C技能树 首页 概览 206136 人正在系统学习中

c++-算法-高精度-高精度减法

我们平时一般利用计算机进行数值计算，有时计算要求计算的精度高，希望计算的数的位数达到上百或者上千，甚至更多。但是由于计算机本身存在的缺陷和硬件问题，在

C++基础算法①——高精度加减法计算

高精度导论、高精度+低精度、高精度+高精度、高精度减法。



AI Code Thunder

关注

👍 0



🌟 3

💬 0