

期中复习之高精度—加法and减法篇

高精度加法：

洛谷P1601

A+B Problem（高精）

题目描述

高精度加法，相当于 $a+b$ problem，不用考虑负数。

输入格式

分两行输入。 $a, b \leq 10^{500}$ 。

输出格式

输出只有一行，代表 $a + b$ 的值。

样例

样例输入

```
1 | 1
2 | 1
```

样例输出

```
1 | 2
```

样例

样例输入

```
1 | 1001
2 | 9099
```

样例输出

```
1 | 10100
```

代码题解：

```
1 | #include <stdio.h>
2 | #include <string.h>
3 | #include <stdlib.h>
4 |
5 | char* add(char*a, char*b);
```

```

6 //做高精度加法的函数。
7 int main()
8 {
9     char a[505];
10    char b[505];
11    //a和b都是10的505次方的量级，按题目需求可以把505改成别的数
12    char* c=(char*)malloc(sizeof(char)*505);
13    //这个就当他是创建了一个名字为c的一个长度为505的char型数组
14    scanf("%s",a);
15    scanf("%s",b);
16    //a和b以字符串的形式给出
17    c = add(a,b);
18
19    printf("%s",c);
20
21    return 0;
22 }
23
24 char* add(char*a, char*b)
25     //本质上是列竖式计算加法的过程
26 {
27     int i,j,k=0;
28     int cin=0;//cin代表是否进位
29     char* c=(char*)malloc(sizeof(char)*505);
30     char* d=(char*)malloc(sizeof(char)*505);
31     //改的时候别忘了把这个505也改了
32     i = strlen(a)-1;
33     j = strlen(b)-1;
34
35     while(i>=0 || j>=0){
36         a[i]=(i<0)?'0':a[i];
37         b[j]=(j<0)?'0':b[j];
38         c[k]=(a[i]-'0'+b[j]-'0'+cin)%10 + '0';
39         //每次计算(a+b+cin)%10的结果
40         cin =(a[i]-'0'+b[j]-'0'+cin)/10;
41         //记录进位情况
42         if(i>=0)
43             i--;
44         if(j>=0)
45             j--;
46         //这里其实有一丢丢不严谨。
47         k++;
48     }
49
50     if(cin==1){
51         c[k]='1';
52         for(i=0;i<=k;i++){
53             d[i]=c[k-i];
54             d[i]='\0';
55             //判断最高位是否有进位的情况
56         }
57     }
58     else{
59         for(i=0;i<=k-1;i++){
60             d[i]=c[k-1-i];
61             d[i]='\0';

```

```
61     }
62     return d;
63 }
```

PS：我已经把高精度加法用函数封装好了，可以当成一个模板复制粘贴就行。期中考试是可以参考本地代码的。

注意适用范围：a和b都是正整数的情况下！

高精度减法：

高精度减法

题目描述

高精度减法。

输入格式

两个整数 a, b （第二个可能比第一个大）。

输出格式

结果（是负数要输出负号）。

样例

样例输入

```
1 2
2 1
```

样例输出

```
1 1
```

提示

- 20% 数据 a, b 在 long long 范围内；
- 100% 数据 $0 < a, b \leq 10^{10086}$ 。

代码题解：

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 char* sub(char* a, char* b);
6 //绝对值减法，且大的减小的
7 char* add(char* a, char* b);
8 //正整数加法
9 int compare(char* a, char* b);
```

```

10 //比较函数
11
12 int main()
13 {
14     char a[11000];
15     char b[11000];
16     char* c = (char*)malloc(sizeof(char)*11000);
17     scanf("%s", a);
18     scanf("%s", b);
19
20     if(a[0]=='-'&&b[0]!='-'){
21         c=add(a+1,b);
22         printf("-%s",c);
23     }
24     //判断负数减正数的情况，结果为负
25     else if (a[0]!='-'&&b[0]=='-'){
26         c=add(a,b+1);
27         printf("%s",c);
28     }
29     //判断正数减负数的情况，结果为正
30     else if (a[0]!='-'&&b[0]!='-'){
31         if(compare(a,b)==1){
32             c=sub(a,b);
33             printf("%s",c);
34         }
35         //a的绝对值大于b的情况，结果为正
36         else if(compare(a,b)==0){
37             printf("0");
38         }
39         else{
40             c = sub(b,a);
41             printf("-%s",c);
42         }
43         //a的绝对值小于b的情况，结果为负
44     }
45     //判断正数减正数的情况
46     else{
47         if(compare(a+1,b+1)==1){
48             c=sub(a+1,b+1);
49             printf("-%s",c);
50         }
51         //a的绝对值大于b的情况，结果为负
52         else if(compare(a+1,b+1)==0){
53             printf("0");
54         }
55         else{
56             c = sub(b+1,a+1);
57             printf("%s",c);
58         }
59         //a的绝对值小于b的情况，结果为正
60     }
61     //判断负数减负数的情况
62     return 0;
63 }
64

```

```

65 int compare(char*a,char*b)//比较a和b代表的数字的大小
66 {
67     int i,j;
68     int len1,len2;
69     len1 = strlen(a);
70     len2 = strlen(b);
71     if(len1 > len2)
72         return 1;
73     else if(len1 < len2){
74         return -1;
75     }
76     else{
77         for(i=0;i<len1;i++){
78             if(a[i]>b[i])
79                 return 1;
80             else if(a[i] < b[i])
81                 return -1;
82         }
83         return 0;
84     }
85 }
86
87 char* sub(char* a,char* b)
88 {
89     int len1,len2;
90     char* c = (char*)malloc(sizeof(char)*11000);
91     len1 = strlen(a);
92     len2 = strlen(b);
93     int cin=0;
94     int i;
95
96     for(i=len2-1;i>=0;i--){
97         b[i+len1-len2]=b[i];
98     }
99     for(i=0;i<len1-len2;i++){
100         b[i]='0';
101     }
102     //因为传进来的a和b已经满足a>b且a和b都大于零。
103     //这一步操作是将a和b对齐，把b的位数补到和a的相同
104
105
106     c[len1]='\0';
107     //注意这里字符串结尾要有'\0'
108
109     //下面这个也是对列竖式的模拟过程。细心一点就好了
110     for(i=len1-1;i>=0;i--){
111         if((a[i] - '0')-cin-(b[i] - '0')>=0){
112             c[i]=(a[i]-'0')-cin-(b[i] - '0') + '0';
113             cin = 0;
114         }
115         else {
116             c[i]=10 + (a[i]-'0')-cin-(b[i]-'0') + '0';
117             cin = 1;
118         }
119     }

```

```

120
121     for(i=0;i<len1&& c[i]==0;i++)
122         ;
123     //去除掉前导0;
124     return c+i;
125 }
126
127 char* add(char*a, char*b)//和上一道题的加法函数一样
128 {
129     int i,j,k=0;
130     int cin=0;
131     char* c=(char*)malloc(sizeof(char)*505);
132     char* d=(char*)malloc(sizeof(char)*505);
133     i = strlen(a)-1;
134     j = strlen(b)-1;
135
136     while(i>=0||j>=0){
137         a[i]=(i<0)?'0':a[i];
138         b[j]=(j<0)?'0':b[j];
139         c[k]=(a[i]-'0'+b[j]-'0'+cin)%10 + '0';
140         cin =(a[i]-'0'+b[j]-'0'+cin)/10;
141         if(i>=0)
142             i--;
143         if(j>=0)
144             j--;
145         k++;
146     }
147
148     if(cin==1){
149         c[k]='1';
150         for(i=0;i<=k;i++)
151             d[i]=c[k-i];
152         d[i]='\0';
153     }
154     else{
155         for(i=0;i<=k-1;i++)
156             d[i]=c[k-1-i];
157         d[i]='\0';
158     }
159     return d;
160 }

```

emmm写的可能会比较啰嗦，但是是对的。也是和上面的一样，用函数封装好了，可以直接用。

注意适用范围：没有前导零的整数的减法（可以有负号是负数）

总结：

高精度计算可以算是模拟题，模拟往往是比较难的，需要很细心并且仔细读题。我之后还会给大家乘法、除法的模拟。大家就当训练思维了。

PS:这些封装好的代码考试都是可以直接copy使用的。