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
```

样例输出

```
2
```

样例

样例输入

```
1001
9099
```

样例输出

```
10100
```

代码题解：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char* add(char*a, char*b);
```

```c
//做高精度加法的函数。
int main()
{
    char a[505];
    char b[505];
    //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
    char* c=(char*)malloc(sizeof(char)*505);
    //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
    scanf("%s",a);
    scanf("%s",b);
//a和b以字符串的形式给出
    c = add(a,b);

    printf("%s",c);

    return 0;
}

char* add(char*a, char*b)
    //本质上是列竖式计算加法的过程
{
    int i,j,k=0;
    int cin=0;//cin代表是否进位
    char* c=(char*)malloc(sizeof(char)*505);
    char* d=(char*)malloc(sizeof(char)*505);
    //改的时候别忘了把这个505也改了
    i = strlen(a)-1;
    j = strlen(b)-1;

    while(i>=0||j>=0){
        a[i]=(i<0)?'0':a[i];
        b[j]=(j<0)?'0':b[j];
        c[k]=(a[i]-'0'+b[j]-'0'+cin)%10 + '0';
        //每次计算(a+b+cin)%10的结果
        cin =(a[i]-'0'+b[j]-'0'+cin)/10;
        //记录进位情况
        if(i>=0)
            i--;
        if(j>=0)
            j--;
        //这里其实有一丢丢不严谨。
        k++;
    }

    if(cin==1){
        c[k]='1';
        for(i=0;i<=k;i++)
            d[i]=c[k-i];
        d[i]='\0';
        //判断最高位是否有进位的情况
    }
    else{
        for(i=0;i<=k-1;i++)
            d[i]=c[k-1-i];
        d[i]='\0';
```

```
    }
    return d;
}
```

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

```
2
1
```

样例输出

```
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：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* sub(char* a,char* b);
//绝对值减法，且大的减小的
char* add(char*a, char*b);
//正整数加法
int compare(char*a,char*b);
```

```c
//比较函数

int main()
{
    char a[11000];
    char b[11000];
    char* c = (char*)malloc(sizeof(char)*11000);
    scanf("%s", a);
    scanf("%s", b);

    if(a[0]=='-'&&b[0]!='-'){
        c=add(a+1,b);
        printf("-%s",c);
    }
    //判断负数减正数的情况，结果为负
    else if (a[0]!='-'&&b[0]=='-'){
        c=add(a,b+1);
        printf("%s",c);
    }
    //判断正数减负数的情况，结果为正
    else if (a[0]!='-'&&b[0]!='-'){
        if(compare(a,b)==1){
            c=sub(a,b);
            printf("%s",c);
        }
        //a的绝对值大于b的情况，结果为正
        else if(compare(a,b)==0){
            printf("0");
        }
        else{
            c = sub(b,a);
            printf("-%s",c);
        }
        //a的绝对值小于b的情况，结果为负
    }
    //判断正数减正数的情况
    else{
        if(compare(a+1,b+1)==1){
            c=sub(a+1,b+1);
            printf("-%s",c);
        }
        //a的绝对值大于b的情况，结果为负
        else if(compare(a+1,b+1)==0){
            printf("0");
        }
        else{
            c = sub(b+1,a+1);
            printf("%s",c);
        }
        //a的绝对值小于b的情况，结果为正
    }
    //判断负数减负数的情况
    return 0;
}

```

```
int compare(char*a,char*b)//比较a和b代表的数字的大小
{
    int i,j;
    int len1,len2;
    len1 = strlen(a);
    len2 = strlen(b);
    if(len1 > len2)
        return 1;
    else if(len1 < len2){
        return -1;
    }
    else{
        for(i=0;i<len1;i++){
            if(a[i]>b[i])
                return 1;
            else if(a[i] < b[i])
                return -1;
        }
        return 0;
    }
}

char* sub(char* a,char* b)
{
    int len1,len2;
    char* c = (char*)malloc(sizeof(char)*11000);
    len1 = strlen(a);
    len2 = strlen(b);
    int cin=0;
    int i;

    for(i=len2-1;i>=0;i--){
        b[i+len1-len2]=b[i];
    }
    for(i=0;i<len1-len2;i++){
        b[i]='0';
    }
    //因为传进来的a和b已经满足a>b且a和b都大于零。
    //这一步操作是将a和b对齐，把b的位数补到和a的相同


    c[len1]='\0';
    //注意这里字符串结尾要有'\0'

    //下面这个也是对列竖式的模拟过程。细心一点就好了
    for(i=len1-1;i>=0;i--){
        if((a[i] - '0')-cin-(b[i] - '0')>=0){
            c[i]=(a[i]-'0')-cin-(b[i] - '0') + '0';
            cin = 0;
        }
        else {
            c[i]=10 + (a[i]-'0')-cin-(b[i]-'0') + '0';
            cin = 1;
        }
    }
```

```
    for(i=0;i<len1&&c[i]==0;i++)
        ;
    //去除掉前导0;
    return c+i;
}

char* add(char*a, char*b)//和上一道题的加法函数一样
{
    int i,j,k=0;
    int cin=0;
    char* c=(char*)malloc(sizeof(char)*505);
    char* d=(char*)malloc(sizeof(char)*505);
    i = strlen(a)-1;
    j = strlen(b)-1;

    while(i>=0||j>=0){
        a[i]=(i<0)?'0':a[i];
        b[j]=(j<0)?'0':b[j];
        c[k]=(a[i]-'0'+b[j]-'0'+cin)%10 + '0';
        cin =(a[i]-'0'+b[j]-'0'+cin)/10;
        if(i>=0)
            i--;
        if(j>=0)
            j--;
        k++;
    }

    if(cin==1){
        c[k]='1';
        for(i=0;i<=k;i++)
            d[i]=c[k-i];
        d[i]='\0';
    }
    else{
        for(i=0;i<=k-1;i++)
            d[i]=c[k-1-i];
        d[i]='\0';
    }
    return d;
}
```

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。