> 创建一个无长度限制的大整型，并实现加减乘除等功能
>
> 参考：
>
> https://blog.csdn.net/qq_36894136/article/details/79074728
>
> https://blog.csdn.net/u013553804/article/details/51175388

大整型头文件

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

/*
    说明：
    num1、num2、result都需在用malloc创建后初始化为0（ASCII码的0）

*/

//为了保证可移植性，对基本类型重命名
typedef int Interger;
typedef char Char;

//设置一个无长度限制的整数存储结构
typedef struct bigint{
     Char *num;      //存储大整型，刚开始是'9'之类的字符，但是经过转换后通过%d输出则为数字了
     Interger sign;      //表示数值的正负，1表示正号，-1表示负号
     Interger digit;     //表示数值的位数，负号不算一位
}BIGINT,*pBIGINT;    //BIGINT是指向结构体的新类型名，pBIGINT是指向结构体的指针变量

//声明函数
void createBigInt(pBIGINT num,int len);
void bigIntTrans(pBIGINT num1);
void bigIntTrim(pBIGINT num1);
void bigIntPrint(pBIGINT result);
Interger bigIntEqual(pBIGINT num1,pBIGINT num2);
void bigIntAdd1(pBIGINT num1,pBIGINT num2,pBIGINT result);
void bigIntSub1(pBIGINT num1,pBIGINT num2,pBIGINT result);
void bigIntSub(pBIGINT num1,pBIGINT num2,pBIGINT result);
void bigIntAdd(pBIGINT num1,pBIGINT num2,pBIGINT result);
void bigIntMul(pBIGINT num1,pBIGINT num2,pBIGINT result);
void bigIntDiv(pBIGINT num1,pBIGINT num2,pBIGINT result,pBIGINT residue);
```

大整型实现

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

```
void createBigInt(pBIGINT num1,int len)
{
     Interger i;//用于遍历
     //创建大整型
     if(!(num1->num=(char *)malloc(sizeof(char) * (len))) )
     {
        printf("内存分配失败！\n");
        exit(0);
     }

     num1->digit = len;

     //初始化为0
     for(i = 0;i < num1->digit;i ++){
        num1->num[i] = 0;
     }
}

//把输入的字符转化为数字，并反转，即num数组的第一位对于个位数(刚开始输入是对应最高位）
void bigIntTrans(pBIGINT num1)
{
    Char temp;  //临时存储位置，用于反转，不用开辟临时数组
    Interger i = 0,len,t;//i用于循环遍历，len用于存储num的长度,t用于保留num1原来的长度
    t = num1->digit;
    len = strlen(num1->num);//得到num1的长度，不计算'\0'，即计算实际长度(包括空格),
    //c系统在用字符数组存储字符串常量时会自动加一个'\0'作为结束符

    /* 这里代码重写了，可以优化，若输入第一位为负号，可以让i自增1
    if(num1->num[0] == '-') //看输入的数据第一位是否是负号
    {
        num1->sign = -1; //设置负号到sign中

        for(i = 1;i < len;i ++){
            num1->num[i] -= '0'; //把每一项减去‘0’，使得按%d输出时可以输出整数
        }

        num1->digit = len - 1; //保存位数
    }else{//是正数，从0位开始
        num1->sign = 1;

        for(i = 0;i < len;i ++){
            num1->num[i] -= '0'; //把每一项减去‘0’，使得按%d输出时可以输出整数
        }

        num1->digit = len; //保存位数
    }
    */
    num1->sign = 1; //先假设数值是正数
    num1->digit = len; //len为数值长度
    if(num1->num[0] == '-') //看输入的数据第一位是否是负号
    {
        num1->sign = -1; //设置负号到sign中
        //注意哦，这时候num1->num[0] == ‘-’,所以，如果进行下面的反转就会出错哦
        for(i = 0;i < num1->digit;i ++){
            num1->num[i] = num1->num[i + 1];//把数组向前移动一位
        }
        num1->digit = len - 1; //保存位数
    }
```

```
    //注意，这里用num1->digit做条件，要是用i<len做条件，
    //那么对于-5这样的来说，num1->num[1] = -48了，多减了一次哦
    for(i = 0;i < num1->digit;i ++){
        num1->num[i] -= '0'; //把每一项减去‘0’，使得按%d输出时可以输出整数
    }

    //i重置0,将数组反转，这里也是，不用i < len/2为条件
    //否则，对于-5来说，反转后num1->digit=1,num1->num[0]=0，会触发除数为0的错误
    for(i = 0;i < num1->digit / 2;i ++){//len / 2对于奇数偶数都成立
        temp = num1->num[i];
        num1->num[i] = num1->num[num1->digit - 1 - i];//这里右边的别忘了是num1-
>digit - 1 - i哦
        num1->num[num1->digit - 1 - i] = temp;//别写成len-1-i
    }

    //调用函数处理最高位后多余的0
    bigIntTrim(num1);

    //清理多余的0后按实际大小重新分配空间
    Char *newbase;
    if(t > num1->digit){
        newbase = (Char*)realloc(num1->num,num1->digit);
        if(newbase == NULL){
            printf("重新分配空间失败！\n");
        }
        num1->num = newbase;
        newbase = NULL;
    }
}


//整理数据，把最高位后面的零不打印输出，即重新设置digit，考虑两种情况：一种是num全部为零；另一
种正常
void bigIntTrim(pBIGINT num1)
{
    Interger i;
    //从最高位开始遍历啦，num1->digit-1
    for(i = num1->digit-1;i >= 0; i --){
        if(num1->num[i] != 0){//注意这里已经减去'0'了，所以这里不能用!='0'，因为ASCII码
不等
            break;
        }
    }
    num1->digit = i + 1;    //记得加1
    if(i < 0){//全部为0的情况
        num1->num[0] = 0;
        num1->digit = 1;
        num1->sign = 1;//0也设为正号吧
    }

}


//把数组num按反序输出，即先输出最高位
void bigIntPrint(pBIGINT result)
{
    bigIntTrim(result);//先调用函数清理一下最高位可能存在的多余的0
```

```
    Interger i;
    //要考虑是否是负数，如果是，输出‘-’
    if(result->sign == -1){
        printf("-");
    }
    for(i = result->digit-1;i >= 0;i --){//包括0的情况
        printf("%d",result->num[i]);
        if(i % 3 ==0 && i != 0) {
            printf(",");//每三位输出一个逗号
        }
    }
}


//比较绝对值的大小，若num1 > num2，返回1；等于返回0；小于返回-1
//比较绝对值时需先调用bigIntTrans及bigIntTrim函数
Interger bigIntEqual(pBIGINT num1,pBIGINT num2)
{
    Interger rs = 0,i = 0;  //rs表示两个数比较的结果，先默认为相等，这样可以省去判断 i小于0的代码
    if(num1->digit > num2->digit){//位数大，则大于
        rs = 1;
    }else if(num1->digit < num2->digit){//位数小，则小于
        rs = -1;
    }else{//位数相等，判断
        for(i = num1->digit-1;i >= 0;i --){//因为位数相等，所以选任意一个的位数赋值给i都行，从高位开始
            if(num1->num[i] > num2->num[i]){
                rs = 1;
                break;
            }else if(num1->num[i] < num2->num[i]){
                rs = -1;
                break;
            }else{
                continue;
            }
        }
    }
    return rs;
}


//根据bigIntEqual函数返回值看是否要调整num1及num2，保证前者大于后者
void bigIntAdd(pBIGINT num1,pBIGINT num2,pBIGINT result)
{
    Interger rs = bigIntEqual(num1,num2);
    if(rs < 1){//如果rs小于1，要把两者互换，保证num1最大
        pBIGINT temp = num1;
        num1 = num2;
        num2 = temp;
        temp = NULL;//使之为空
    }

    if(num1->sign * num2->sign > 0){//同号相加，包括-7+(-5),7+5等
        bigIntAdd1(num1,num2,result);
    }else{//异号相加，也可以认为是相减
    //所以这时候不用考虑3-（-5），-7-7这种情况了，因为把减法转化为加法后，他们符合同号相加，被筛掉了
```

```
        if(rs == 0){//如果等于0，那么相减就为0了哦
            result->digit = 1;
            result->num[0] = 0;
            result->sign = 1;
            return ;
        }
        bigIntSub1(num1,num2,result);
    }
}

//实现同号相加
void bigIntAdd1(pBIGINT num1,pBIGINT num2,pBIGINT result)
{
    result->sign = num1->sign;//结果的正负跟num1一致，因为是同号相加
    //先设结果的位数为num1的位数+1，即使有多余的0，最后一行代码会纠正的
    result->digit = num1->digit + 1;
    Interger i; //i循环遍历
    for(i = 0;i < num1->digit;i ++){//注意哦，这里i<num1->digit不用减去1，要不然出错
    //这里直接以num1的digit为条件吧，开辟空间时num1与num2一样大，然后都初始化为0
        result->num[i] += num1->num[i] + num2->num[i];//记得加上进位
        if(result->num[i] >= 10){
            //进位也不一定是进位1，而且用+=更准确
            result->num[i+1] += result->num[i] / 10;
            result->num[i] %= 10;
        }else{
            result->num[i+1] = 0; //如果无进位，也保存在result的num数组的下一位，赋值0
        }
    }
    bigIntTrim(result);
}

//把减号转化为加号，减去一个数等于加上这个数的相反数
void bigIntSub(pBIGINT num1,pBIGINT num2,pBIGINT result)
{
    /* 以下代码单独写会出现问题，其实减数应该转换为加法，所以num2的sign应该取反，然后因为都是
加法了
       所以可以调用add函数处理

    Interger rs = bigIntEqual(num1,num2);
    if(rs < 1){//如果rs小于1，要把两者互换，保证num1最大
        pBIGINT temp = num1;
        num1 = num2;
        num2 = temp;
        temp = NULL;//使之为空
    }

    if(num1->sign * num2->sign < 0){//对-1-5或者5-（-3）这种的，转化为用add1函数运算
        bigIntAdd1(num1,num2,result); //这里不妥，如果是3-(-5)，那么调换后-5为num1，则
结果为负数？
        //但是如果设置num1为正数又不对，比如-7-7结果是负数……
    }else{
        if(rs == 0){//如果等于0，那么相减就为0了哦
            result->digit = 1;
            result->num[0] = 0;
            result->sign = 1;
            return ;
        }
        bigIntSub1(num1,num2,result);
```

```
    }
    */
    num2->sign = -1 * num2->sign;    //减去一个数等于加上这个数的相反数
    bigIntAdd(num1,num2,result);
}

//执行5-3之类的运算
void bigIntSub1(pBIGINT num1,pBIGINT num2,pBIGINT result)
{
    result->sign = num1->sign;  //结果的符号取决于num1
    result->digit = num1->digit;//先设结果的位数为num1的位数，即使有多余的0，最后一行代码会纠正的
    Interger i,k = 0;   //i循环遍历,k保留进位
    for(i = 0;i < num1->digit;i ++){
        //减去result->num[i]是因为用它来存储借位
        result->num[i] = num1->num[i] - num2->num[i] - k;
        if(result->num[i] < 0){
            result->num[i] += 10;
            k = 1;//注意这里是1，因为上面是减去k
        }else{
            k = 0;  //表示没有借位
        }
    }
    bigIntTrim(result);
}

//执行乘法运算
//乘法的原理是让乘数的每一位乘以被乘数，然后加起来；
//两个乘数相乘最大的位数为二者之和，如99*99
void bigIntMul(pBIGINT num1,pBIGINT num2,pBIGINT result)
{
    result->sign = num1->sign * num2->sign;
    result->digit = num1->digit + num2->digit;
    Interger i,j,carry,pos; //i循环遍历num2的位数，k循环遍历num1的位数
    Char temp;
    for(i = 0;i < num2->digit;i ++){

        for(j = 0;j < num1->digit;j ++){

            //这里的+=是因为会有进位的可能，这里巧妙的用i+j存储结果，是因为i,j都是从0开始
            //这样就可以巧妙地在乘数第二位乘时结果从十位开始
            result->num[i+j] = result->num[i+j] +  num1->num[j] * num2->num[i];

            if(result->num[i+j] >= 10){
                 //注意，不一定是进一位哦，而且要是+=才行，因为可能有两次进位，如99*99
                result->num[i+j+1] += result->num[i+j] / 10;
                result->num[i+j] %= 10;
            }/*原来是这行代码错了，不应该加的，要不然乘数可能出错，因为可能会把已经有的进位清零的
            else{
                result->num[i+j+1] = 0; //其实不用写这一行代码也行，因为初始化为0了
            }*/

        }

        /* 下面这个是一位CSDN大佬的代码，可能思路更清晰

        carry=0;                                    //清除进位
```

```
        //被乘数的每一位
        for(j=0;j<num1->digit;j++)
        {
            //相乘并加上进位
            temp=num2->num[i] * num1->num[j]+carry;
             //计算进位carry
            carry =temp/10;
            //计算当前位的值
            temp=temp%10;
            pos=i+j;
            //计算结果累加到临时数组中
            result->num[pos]+=temp;
            carry=carry+result->num[pos]/10;            //计算进位
            result->num[pos]=result->num[pos]%10;
        }
        if(carry>0)
        {
            result->num[i+j]=carry;                  //加上最高位进位
            result->digit=i+j+1;                    //保存结果位数
        }else
            result->digit=i+j;                     //保存结果位数
        */
    }
    bigIntTrim(result);
}


//执行除法运算
//从被除数最高位开始，看能不能除以除数，如果不能向后借一位，依次类推
//余数用一个pBIGINT类型记录，避免是大整型的数吧
//存在被除数小于除数的情况，这时余数为被除数
void bigIntDiv(pBIGINT num1,pBIGINT num2,pBIGINT result,pBIGINT
residue)//residue表示余数的意思
{
    result->sign = num1->sign * num2->sign;//确定符号

    if(num2->num[0] == 0 && num2->digit == 1){//除数为0，不允许
        printf("除数不能为0!\n");
        exit(0);
    }
    //i循环遍历;j遍历余数位数；k保留试商结果，m保留商的位数 ;temp用于商反转时
    Interger i,j,k = 0,m = 0,temp;
    for(i = num1->digit-1;i >= 0;i --){
        residue->digit = num2->digit + 1; //余数的位数先设置为num2-
>digit+1,10000/9999的情况

        for(j = residue->digit - 1;j > 0;j --){//这里的判断条件如果是j >=0，那么
residue->num[j-1]会溢出
            residue->num[j] = residue->num[j-1]; //余数提一位，因为余数最低位应该是个
位
        }
        residue->num[0] = num1->num[i];//把num1的下一位赋值给residue
        bigIntTrim(residue);    //整理余数

        //这一块想除行不通，因为是个数组而不是一个数，所以只能不断地减
        while(bigIntEqual(residue,num2) >= 0){//余数比除数大，那么相减知道比除数小，减
的次数即为商
            bigIntSub1(residue,num2,residue);//拿余数减去除数，差的结果保存在余数中
```

```
            k ++;//计算减的次数
        }
        result->num[m ++] = k;  //等下记得反转
        k = 0; //商归0
    }
    result->digit = m;  //保存结果的位数
    for(i = 0;i < m / 2;i ++){
        temp = result->num[i];
        result->num[i] = result->num[m-1-i];
        result->num[m-1-i] = temp;
    }
    bigIntTrim(result);
    bigIntTrim(residue);
}
```

大整型测试

```
#include <stdio.h>
#include <stdlib.h>
#include "bigint.h"
#include <string.h>
#include <windows.h>

int menu(char op,int flag,int len,pBIGINT num1,pBIGINT num2,pBIGINT
result,pBIGINT residue);

int main()
{
    system("color 4F");//控制小窗口颜色

    //参与运算的数，结果，余数
    //pBIGINT num1,num2,result,residue;
    //如果定义上面这行代码在运行到createBigInt(num1,len + 1)函数的malloc分配时出错
    //Program received signal SIGSEGV, Segmentation fault。
    //即使初始化为NULL也没用，原因在于初始化为NULL后，再在createBigInt那里给num分配空间
    //就有问题了，因为访问不存在的内存；同理，如果是上面那行代码估计在运行时没有分配一个
    //结构体的内存，即成员变量可能不存在
    //num1 = num2 = result = residue = NULL;
    //所以先声明为结构体类型，再用&得到其地址
    BIGINT num1,num2,result,residue;
    int i=0,len,flag = 1;//flag用于标记输入的数值是否大于所说的最大数的位数，0表示大于
    char op;
    printf("输入最大数的位数：");
    scanf("%d",&len);
    //创建大整型
    createBigInt(&num1,len + 1);
    createBigInt(&num2,len + 1);
    createBigInt(&result,2 * len + 1);

    printf("\n选择大整数的运算（+,-,*,/）： ");
    fflush(stdin); //清除缓冲区，如果没有这行代码，会把5直接赋值给op
    scanf("%c",&op);
    //调用菜单函数
    flag = menu(op,flag,len,&num1,&num2,&result,&residue);
```

```
    if(flag == 1)
        bigIntPrint(&result);
    if(residue.digit > 0 && op == '/'){
        printf("......");
        bigIntPrint(&residue);
    }
    printf("\n\n");

    //释放内存
    free(num1.num);
    free(num2.num);
    free(result.num);
    free(residue.num);
    num1.num = num2.num = result.num = residue.num = NULL;

    system("pause");//解决.exe文件一闪而过
    return 0;
}

//把选择独立成一个函数 ,所以num1之类的要变成指针才行
int menu(char op,int flag,int len,pBIGINT num1,pBIGINT num2,pBIGINT
result,pBIGINT residue){
    int i;//循环遍历
    switch(op)
    {
        case '+':
            printf("\n输入被加数：");
            scanf("%s",num1->num);
            printf("\n输入加数：");
            scanf("%s",num2->num);
            if(strlen(num1->num) > len || strlen(num2->num) > len){
                flag = 0;
                printf("\n您输入的数值超过之前所定义的最大位数！\n");
                break;
            }
            printf("\n%s + ",num1->num);
            //若为负数，输出带括号
            if(num2->num[0] == '-') {
                printf("（ %s ） = ",num2->num);
            }else{
                printf("%s = ",num2->num);
            }
            bigIntTrans(num1);
            bigIntTrans(num2);
            bigIntAdd(num1,num2,result);      //加法
            break;
        case '-':
            printf("\n输入被减数：");
            scanf("%s",num1->num);
            printf("\n输入减数：");
            scanf("%s",num2->num);
            if(strlen(num1->num) > len || strlen(num2->num) > len){
                flag = 0;
                printf("\n您输入的数值超过之前所定义的最大位数！\n");
                break;
            }
            printf("\n%s - ",num1->num);
```

```
            if(num2->num[0] == '-') {
                printf("（ %s ） = ",num2->num);
            }else{
                printf("%s = ",num2->num);
            }
            bigIntTrans(num1);
            bigIntTrans(num2);
            bigIntSub(num1,num2,result);       //减法
            break;
        case '*':
            printf("\n输入被乘数：");
            scanf("%s",num1->num);
            printf("\n输入乘数：");
            scanf("%s",num2->num);
            if(strlen(num1->num) > len || strlen(num2->num) > len){
                flag = 0;
                printf("\n您输入的数值超过之前所定义的最大位数！\n");
                break;
            }
            printf("\n%s * ",num1->num);
            if(num2->num[0] == '-') {
                printf("（ %s ） = ",num2->num);
            }else{
                printf("%s = ",num2->num);
            }
            bigIntTrans(num1);
            bigIntTrans(num2);
            bigIntMul(num1,num2,result);       //乘法
            break;
        case '/':
            printf("\n输入被除数：");
            scanf("%s",num1->num);
            printf("\n输入除数：");
            scanf("%s",num2->num);
            if(strlen(num1->num) > len || strlen(num2->num) > len){
                flag = 0;
                printf("\n您输入的数值超过之前所定义的最大位数！\n");
                break;
            }
            printf("\n%s / ",num1->num);
            if(num2->num[0] == '-') {
                printf("（ %s ） = ",num2->num);
            }else{
                printf("%s = ",num2->num);
            }
            bigIntTrans(num1);
            bigIntTrans(num2);
            if(num2->digit==1 && num2->num[0]==0){        //大整数为0

                printf("除数不能为0！\n");
                exit(0);

            } else{
                 createBigInt(residue,num2->digit + 2);
                 residue->sign = 1;
                 bigIntDiv(num1,num2,result,residue);     //除法
            }
            break;
```

```
        default:
            printf("\n不存在该运算！\n");
            exit(0);
    }
    return flag;//别忘了返回flag，不知道为啥不返回貌似不出错
}
```