high precision arithmetic（四位一格）

高精输入

参数:数值地址（数组名）

返回值:（int型）高精位数（四位一格）

//阿蓉

int high\_precision\_scanf(int\*p){

char c[40000];

scanf("%s",c);

int l,t,i,j;

l=strlen(c);

if(c[0]=='-'){

l--;

t=(l-1)/4;

l%=4;

if(!l)l=4;

\*(p+t)=0;

for(i=1;i<=l;i++){

\*(p+t)=\*(p+t)\*10+(c[i]-'0');

}

for(j=t-1;j>=0;j--){

\*(p+j)=0;

for(int k=0;k<=3;k++){

\*(p+j)=\*(p+j)\*10+(c[i]-'0');

i++;

}

}

\*(p+t)=-\*(p+t);

return t+1;

}

else{

t=(l-1)/4;

l%=4;

if(!l)l=4;

\*(p+t)=0;

for(i=0;i<l;i++){

\*(p+t)=\*(p+t)\*10+(c[i]-'0');

}

for(j=t-1;j>=0;j--){

\*(p+j)=0;

for(int k=0;k<=3;k++){

\*(p+j)=\*(p+j)\*10+(c[i]-'0');

i++;

}

}

return t+1;

}

}

高精输出

参数:数组地址（数组名），数值高精位数（四位一格）

返回值:无

//阿蓉

void high\_precision\_printf(int\*p,int t){

printf("%d",\*(p+t-1));

for(int i=t-2;i>=0;i--){

printf("%04d",\*(p+i));

}

return ;

}

高精加减（一体捆绑）

参数:存答案数组名（不可与合数数组重合）（请先清0），数值名1，数值1高精位数，数组名2，数组2高精位数

返回值:答案高精长度

//阿蓉

int high\_precision\_add(int \*pc,int \*pa,int la,int \*pb,int lb){

if(\*(pa+la-1)\*\*(pb+lb-1)<0){

if(\*(pa+la-1)<0){

\*(pa+la-1)=-\*(pa+la-1);

int m;

m=high\_precision\_subtraction(pc,pb,lb,pa,la);

\*(pa+la-1)=-\*(pa+la-1);

return m;

}

else{

\*(pb+lb-1)=-\*(pb+lb-1);

int m;

m=high\_precision\_subtraction(pc,pa,la,pb,lb);

\*(pb+lb-1)=-\*(pb+lb-1);

return m;

}

}

else{

int a=0;

if(\*(pa+la-1)<0){

a=1;

\*(pa+la-1)=-\*(pa+la-1);

\*(pb+lb-1)=-\*(pb+lb-1);

}

const int N=10000;

int i,n,m;

n=la>lb?lb:la;

for(i=0;i<n;i++){

\*(pc+i)+=\*(pa+i)+\*(pb+i);

m=i;

while(\*(pc+m)/N!=0){

\*(pc+m+1)+=\*(pc+m)/N;

\*(pc+m)%=N;

m++;

}

}

for(i=la;i<lb;i++){

\*(pc+i)+=\*(pb+i);

m=i;

while(\*(pc+m)/N!=0){

\*(pc+m+1)+=\*(pc+m)/N;

\*(pc+m)%=N;

m++;

}

}

for(i=lb;i<la;i++){

\*(pc+i)+=\*(pa+i);

m=i;

while(\*(pc+m)/N!=0){

\*(pc+m+1)+=\*(pc+m)/N;

\*(pc+m)%=N;

m++;

}

}

if(a){

\*(pa+la-1)=-\*(pa+la-1);

\*(pb+lb-1)=-\*(pb+lb-1);

\*(pc+m)=-\*(pc+m);

}

return m+1;

}

}

//阿蓉

int high\_precision\_subtraction(int \*pc,int \*pa,int la,int \*pb,int lb){

if(\*(pa+la-1)\*\*(pb+lb-1)<0){

\*(pb+lb-1)=-\*(pb+lb-1);

int m;

m=high\_precision\_add(pc,pa,la,pb,lb);

\*(pb+lb-1)=-\*(pb+lb-1);

return m;

}

else{

int m,a=0,c=0,i;

const int N=10000;

if(\*(pa+la-1)<0){

a=1;

\*(pa+la-1)=-\*(pa+la-1);

\*(pb+lb-1)=-\*(pb+lb-1);

}

if(la<lb){

int\*pc;

c=1;

pc=pa;

pa=pb;

pb=pc;

la^=lb;

lb^=la;

la^=lb;

}

for(i=la-1;i>=lb;i--){

\*(pc+i)=\*(pa+i);

}

for(i=lb-1;i>=0;i--){

\*(pc+i)=\*(pa+i)-\*(pb+i);

m=i;

while(\*(pc+m)<0){

\*(pc+m)+=N;

\*(pc+(++m))-=1;

}

}

m=la-1;

while(!\*(pc+m))m--;

if(c)\*(pc+m)=-\*(pc+m);

if(a){

\*(pa+la-1)=-\*(pa+la-1);

\*(pb+lb-1)=-\*(pb+lb-1);

\*(pc+m)=-\*(pc+m);

}

return m+1;

}

}

高精乘法

参数:存答案数组名（不可与合数数组重合）（请先清0），数值名1，数值1高精位数，数组名2，数组2高精位数

返回值:答案高精位数

//阿蓉

int high\_precision\_multiply(int \*pc,int \*pa,int la,int \*pb,int lb){

const int N=10000;

int i,j,m,a=0,b=0;

if(\*(pa+la-1)<0){

a=1;

\*(pa+la-1)=-\*(pa+la-1);

}

if(\*(pb+lb-1)<0){

b=1;

\*(pb+lb-1)=-\*(pb+lb-1);

}

for(i=0;i<la;i++){

for(j=0;j<lb;j++){

\*(pc+i+j)+=\*(pa+i)\*\*(pb+j);

}

m=i;

while(\*(pc+m)%N!=0){

\*(pc+(++m))+=\*(pc+(m-1))/N;

\*(pc+(m-1))%=N;

}

}

if(a){

\*(pa+la-1)=-\*(pa+la-1);

\*(pc+m-1)=-\*(pc+m-1);

}

if(b){

\*(pb+lb-1)=-\*(pb+lb-1);

\*(pc+m-1)=-\*(pc+m-1);

}

return m;

}

排序

归并（指针，手动sort）

参数：数组首地址（如：想从a[n]开始排序，就输a+n。a[0]就是a），想要排序的长度。

返回值：无

//阿蓉

void sort(int\*p,int n){

if(n==1)return;

int mid=n>>1,a[n],i,j,l=0;

sort(p,mid);

sort(p+mid,n-mid);

i=0;j=mid;

while(i<mid&&j<n){

if(\*(p+i)<=\*(p+j))a[l++]=\*(p+(i++));//若要降序排列，把这行的“<”改“>”

else a[l++]=\*(p+(j++));

}//核心循环，若要实现其他排序，在这里进行更改

while(i<mid)a[l++]=\*(p+(i++));

while(j<n)a[l++]=\*(p+(j++));

for(i=0;i<n;i++){

\*(p+i)=a[i];

}

return;

}

幂运算

fastpower（快速幂）

参数：底数，指数

返回值：幂运算结果

//阿蓉

long long fastpower(long long m,int n){

long long ans=1;

while(n){

if(n&1)ans=ans\*m;

m=m\*m;

n>>=1;

}

return ans;

}

矩阵算术

矩阵乘法

参数：答案承接数组名，前一个矩阵名，前矩阵的行数，前矩阵的列数，后一个矩阵名，后矩阵行数，后矩阵列数

返回值：无

说明

保证输入时从0行0列开始输入

记得更新全局常数N

满足矩阵乘法数学要求

//阿蓉

//此函数的三个数组名允许重复，即mat\_multiply(a,a,5,5,a,5,5);合法，结果是a变成原来a矩阵的平方

const int N=101;//按要求修改N的值，使得N等于调用数组的列数

void mat\_multiply(int c[][N],int a[][N],int ha,int la,int b[][N],int hb,int lb){

if(la!=hb)return;//矩阵乘法务必保证la=hb

if(c==a){

int d[ha][N];

for(int i=0;i<ha;i++)

for(int j=0;j<la;j++)

d[i][j]=a[i][j];

a=d;

if(c==b)b=d;

}

else if(c==b){

int d[hb][N];

for(int i=0;i<hb;i++)

for(int j=0;j<lb;j++)

d[i][j]=b[i][j];

b=d;

}

for(int i=0;i<ha;i++){

for(int j=0;j<lb;j++){

c[i][j]=0;

for(int k=0;k<la;k++){

c[i][j]=c[i][j]+a[i][k]\*b[k][j];

}

}

}

return;

}