数学模板

数学模板

- 素数打表
- 求排列组合数
- 行列式计算
- Ronberg算法计算积分
- 组合序列
- 最大公约数、最小公倍数
- 任意进制转换
- 大数问题
  - 大数阶乘
  - 大数加法
  - 大数减法(未处理负数情况)
  - 大数乘法(大数乘小数)
  - 大数乘法(大数乘大数)
  - 大数比较

素数打表

```
/*
语法：primetable(n,prime)

头文件：stdio.h string.h

参数：
m：求小于等于n的素数的个数中的n
prime：存素数的数组
返回值：null

*/

/*求小于等于n的素数的个数*/
int prime[100001];//存素数
void primetable(int n,int prime[])
{
```

```
    int cnt = 0;
    bool vis[100001];//保证不做素数的倍数
    memset(vis, false, sizeof(vis));//初始化
    memset(prime, 0, sizeof(prime));
    for(int i = 2; i <= n; i++)
    {
        if(!vis[i])//不是目前找到的素数的倍数
        prime[cnt++] = i;//找到素数~
        for(int j = 0; j<cnt && i*prime[j]<=n; j++)
        {
            vis[i*prime[j]] = true;//找到的素数的倍数不访问
            if(i % prime[j] == 0) break;//关键！！！
        }
    }
    printf("%d\n", cnt);
}
```

求排列组合数

```
/*
语法：result=P(long n,long m); / result=long C(long n,long m);

参数：
m：排列组合的上系数
n：排列组合的下系数
返回值：排列组合数

注意：
符合数学规则：m<=n
*/
long P(long n,long m)
{
    long p=1;
    while(m!=0)
        {p*=n;n--;m--;}
    return p;
}

long C(long n,long m)
{
    long i,c=1;
    i=m;
```

```
    while(i!=0)
        {c*=n;n--;i--;}
    while(m!=0)
        {c/=m;m--;}
    return c;
}
```

行列式计算

```
/*
语法：result=js(int s[][],int n)

参数：
s[][]：行列式存储数组
n：行列式维数，递归用
返回值：行列式值

注意：函数中常数N为行列式维度，需自行定义
*/

int s[][N],n;
int js(s,n) {
    int z,j,k,r,total=0;
    int b[N][N];
    if(n>2)
        {
        for(z=0;z<n;z++)
            {
            for(j=0;j<n-1;j++)
                 for(k=0;k<n-1;k++){
                        if(k>=z) b[j][k]=s[j+1][k+1];
                        else b[j][k]=s[j+1][k];
                        }
            if(z%2==0) r=s[0][z]*js(b,n-1);
            else r=(-1)*s[0][z]*js(b,n-1);
            total=total+r;
            }
        }
    else if(n==2)
       total=s[0][0]*s[1][1]-s[0][1]*s[1][0];
    return total;
}
```

Ronberg算法计算积分

```
/*
语法：result=integral(double a,double b);

头文件：math.h

参数：
a：积分上限
b：积分下限
function f：积分函数
返回值：f在（a,b）之间的积分值

注意：
function f(x)需要自行修改，程序中用的是sina(x)/x
默认精度要求是1e-5
*/
double f(double x)
{
    return sin(x)/x; //在这里插入被积函数
}

double integral(double a,double b)
{
    double h=b-a;
    double t1=(1+f(b))*h/2.0;
    int k=1;
    double r1,r2,s1,s2,c1,c2,t2;
loop:
    double s=0.0;
    double x=a+h/2.0;
    while(x<b)
        {
        s+=f(x);
        x+=h;
        }
    t2=(t1+h*s)/2.0;
    s2=t2+(t2-t1)/3.0;
    if(k==1)
      {
        k++;h/=2.0;t1=t2;s1=s2;
```

```
            goto loop;
            }
        c2=s2+(s2-s1)/15.0;
        if(k==2){
            c1=c2;k++;h/=2.0;
            t1=t2;s1=s2;
            goto loop;
            }
        r2=c2+(c2-c1)/63.0;
        if(k==3){
            r1=r2; c1=c2;k++;
            h/=2.0;
            t1=t2;s1=s2;
            goto loop;
            }
        while(fabs(1-r1/r2)>1e-5){
            r1=r2;c1=c2;k++;
            h/=2.0;
            t1=t2;s1=s2;
            goto loop;
            }
        return r2;
}
```

组合序列

```
/*
语法：m_of_n(int m, int n1, int m1, int* a, int head)

参数：
m：组合数C的上参数
n1：组合数C的下参数
m1：组合数C的上参数，递归之用
*a：1～n的整数序列数组
head：头指针

返回值：null

注意：*a需要自行产生
初始调用时，m=m1、head=0
调用例子：求C(m,n)序列：m_of_n(m,n,m,a,0);
```

```
*/

void m_of_n(int m, int n1, int m1, int* a, int head)
{
    int i,t;
    if(m1<0 || m1>n1) return;

    if(m1==n1)
        {
        for(i=0;i<m;i++) cout<<a[i]<<' '; // 输出序列
        cout<<'\n';
        return;
        }
    m_of_n(m,n1-1,m1,a,head); // 递归调用
    t=a[head];a[head]=a[n1-1+head];a[n1-1+head]=t;
    m_of_n(m,n1-1,m1-1,a,head+1); // 再次递归调用
    t=a[head];a[head]=a[n1-1+head];a[n1-1+head]=t;
}
```

最大公约数、最小公倍数

```
/*
语法：resulet=hcf(int a,int b)、result=lcd(int a,int b)

参数：
a：int a，求最大公约数或最小公倍数
b：int b，求最大公约数或最小公倍数

返回值：返回最大公约数（hcf）或最小公倍数（lcd）

注意：lcd 需要连同 hcf 使用
*/

int hcf(int a,int b)
{
    int r=0;
    while(b!=0)
        {
        r=a%b;
        a=b;
        b=r;
        }
```

```
    return(a);
}

lcd(int u,int v,int h)
{
    return(u*v/h);
}
```

任意进制转换

```
/*
语法：conversion(char s1[],char s2[],long d1,long d2);

参数：
s[]：原进制数字，用字符串表示
s2[]：转换结果，用字符串表示
d1：原进制数
d2：需要转换到的进制数

返回值：
null

注意：
高于9的位数用大写'A'～'Z'表示，2～16位进制通过验证
*/
void conversion(char s[],char s2[],long d1,long d2)
{
    long i,j,t,num;
    char c;
    num=0;
    for (i=0;s[i]!='\0';i++)
        {
        if (s[i]<='9'&&s[i]>='0') t=s[i]-'0'; else t=s[i]-'A'+10;
        num=num*d1+t;
        }
    i=0;
    while(1)
        {
        t=num;
        if (t<=9) s2[i]=t+'0'; else s2[i]=t+'A'-10;
```

```
        num/=d2;
        if (num==0) break;
        i++;
        }
    for (j=0;j<i/2;j++)
        {c=s2[j];s2[j]=s[i-j];s2[i-j]=c;}
    s2[i+1]='\0';
}
```

大数问题

大数阶乘

```
/*
语法：int result=factorial(int n);

头文件：math.h stdio.h

参数：
n：n 的阶乘
返回值：阶乘结果的位数

注意：
本程序直接输出n!的结果，需要返回结果请保留long a[]
*/

int factorial(int n){
      long a[10000];
      int i,j,l,c,m=0,w;
      a[0]=1;
      for(i=1;i<=n;i++)
          {
          c=0;
          for(j=0;j<=m;j++)
              {
              a[j]=a[j]*i+c;
              c=a[j]/10000;
              a[j]=a[j]%10000;
          }
          if(c>0) {m++;a[m]=c;}
      }
```

```
        w=m*4+log10((double)a[m])+1;
        printf("%ld",a[m]);
        for(i=m-1;i>=0;i--) printf("%4.4ld",a[i]);
        return w;
 }
```

大数加法

```
/*
      语法：add(char a[],char b[],char s[]);

      头文件：string.h

      参数：
      a[]：被加数，用字符串表示，位数不限
      b[]：加数，用字符串表示，位数不限
      s[]：结果，用字符串表示
      返回值：null

      注意：
      空间复杂度为 o(n^2)
*/

void add(char a[],char b[],char back[]){
          int i,j,k,up,x,y,z,l;
          char *c;
          if (strlen(a)>strlen(b))
          l=strlen(a)+2;
          else l=strlen(b)+2;
          c=(char *) malloc(l*sizeof(char));
          i=strlen(a)-1;
          j=strlen(b)-1;
          k=0;up=0;
          while(i>=0||j>=0)
              {
                  if(i<0) x='0'; else x=a[i];
                  if(j<0) y='0'; else y=b[j];
                  z=x-'0'+y-'0';
                  if(up) z+=1;
                  if(z>9) {up=1;z%=10;} else up=0;
                  c[k++]=z+'0';
                  i--;j--;
```

```
            }
        if(up) c[k++]='1';
        i=0;
        c[k]='\0';
        for(k-=1;k>=0;k--)
            back[i++]=c[k];
        back[i]='\0';
    }
```

大数减法(未处理负数情况)

```
/*
  语法：sub(char s1[],char s2[],char t[]);

  头文件：string.h

     参数：
      s1[]：被减数，用字符串表示，位数不限
      s2[]：减数，用字符串表示，位数不限
      t[]：结果，用字符串表示
      返回值：null

      注意：
      默认s1>=s2，程序未处理负数情况(倒过来加符号)

*/
void sub(char s1[],char s2[],char t[])
    {
        int i,l2,l1,k;
        l2=strlen(s2);l1=strlen(s1);
        t[l1]='\0';l1--;
        for (i=l2-1;i>=0;i--,l1--)
            {
            if (s1[l1]-s2[i]>=0)
                t[l1]=s1[l1]-s2[i]+'0';
            else
                {
                t[l1]=10+s1[l1]-s2[i]+'0';
                s1[l1-1]=s1[l1-1]-1;
                }
            }
        k=l1;
```

```
        while(s1[k]<0) {s1[k]+=10;s1[k-1]-=1;k--;}
        while(l1>=0) {t[l1]=s1[l1];l1--;}
    loop:
        if (t[0]=='0')
            {
            l1=strlen(s1);
            for (i=0;i<l1-1;i++) t[i]=t[i+1];
            t[l1-1]='\0';
            goto loop;
            }
        if (strlen(t)==0){t[0]='0';t[1]='\0';}
    }
```

大数乘法(大数乘小数)

```
/*
  语法：mult(char c[],char t[],int m);

  头文件：string.h

     参数：
      c[]：被乘数，用字符串表示，位数不限
      t[]：结果，用字符串表示
      m：乘数，限定10以内
      返回值：null
*/

void mult(char c[],char t[],int m)
    {
        int i,l,k,flag,add=0;
        char s[100];
        l=strlen(c);
        for (i=0;i<l;i++)
            s[l-i-1]=c[i]-'0';
        for (i=0;i<l;i++)
               {
               k=s[i]*m+add;
               if (k>=10) {s[i]=k%10;add=k/10;flag=1;} else
    {s[i]=k;flag=0;add=0;}
               }
        if (flag) {l=i+1;s[i]=add;} else l=i;
```

```
            for (i=0;i<l;i++)
                t[l-1-i]=s[i]+'0';
            t[l]='\0';
        }
```

大数乘法(大数乘大数)

```
/*
  语法：mult(char a[],char b[],char s[]);

  头文件：string.h

      参数：
       a[]：被乘数，用字符串表示，位数不限
       b[]：乘数，用字符串表示，位数不限
       t[]：结果，用字符串表示
       返回值：null

       注意：
       空间复杂度为 o(n^2)
*/

void mult(char a[],char b[],char s[])
       {
            int i,j,k=0,alen,blen,sum=0,res[65][65]={0},flag=0;
            char result[65];
            alen=strlen(a);blen=strlen(b);
            for (i=0;i<alen;i++)
            for (j=0;j<blen;j++) res[i][j]=(a[i]-'0')*(b[j]-'0');
            for (i=alen-1;i>=0;i--)
                {
                    for (j=blen-1;j>=0;j--) sum=sum+res[i+blen-j-1]
[j];
                    result[k]=sum%10;
                    k=k+1;
                    sum=sum/10;
                }
            for (i=blen-2;i>=0;i--)
                {
                    for (j=0;j<=i;j++) sum=sum+res[i-j][j];
                    result[k]=sum%10;
                    k=k+1;
```

```
                sum=sum/10;
            }
        if (sum!=0) {result[k]=sum;k=k+1;}
        for (i=0;i<k;i++) result[i]+='0';
        for (i=k-1;i>=0;i--) s[i]=result[k-1-i];
        s[k]='\0';
        while(1)
            {
            if (strlen(s)!=strlen(a)&&s[0]=='0')
                strcpy(s,s+1);
            else
                break;
            }
    }
```

大数比较

```
/*
  语法：int compare(char a[],char b[]);

      参数：
      a[]：被比较数，用字符串表示，位数不限
      b[]：比较数，用字符串表示，位数不限

      返回值：
      0    a<b
      1    a>b
      2    a=b
*/

int compare(char a[], char b[])
{
    int lena=strlen(a);
    int lenb=strlen(b);
    if(lena>lenb)
        return 1;
    else if(lena<lenb)
        return 0;
    for(int i=0;i<lena;i++)
    {
        if(a[i]>b[i])
            return 1;
```

```
            else if(a[i]<b[i])
                return 0;
        }
        return 2;
    }
```