C 语言常用算法归纳

应当掌握的一般算法

一、基本算法：

交换、累加、累乘

二、非数值计算常用经典算法：

穷举、排序（冒泡，选择）、查找（顺序即线性）

三、数值计算常用经典算法：

级数计算（直接、简接即递推）、一元非线性方程求根（牛顿迭代法、二分法）、定积分计算（矩形法、梯形法）、矩阵转置

四、其他：

迭代、进制转换、字符处理（统计、数字串、字母大小写转换、加密等）、整数各数位上数字的获取、辗转相除法求最大公约数（最小公倍数）、求最值、判断素数（各种变形）、数组元素的插入（删除）、二维数组的其他典型问题（方阵的特点、杨辉三角形）

详细讲解

一、基本算法

1．交换（两量交换借助第三者）

例 1、任意读入两个整数，将二者的值交换后输出。

```
   main()
{int a,b,t;
 scanf("%d%d",&a,&b);
 printf("%d,%d\n",a,b);
 t=a;   a=b;   b=t;
 printf("%d,%d\n",a,b);}
```

【解析】程序中**加粗**部分为算法的核心，如同交换两个杯子里的饮料，必须借助第三个空杯子。

假设输入的值分别为 3、7，则第一行输出为 3，7；第二行输出为 7，3。

其中 t 为中间变量，起到“空杯子”的作用。

注意：三句赋值语句赋值号左右的各量之间的关系！

【应用】

例 2、任意读入三个整数，然后按从小到大的顺序输出。

```
main()
{int a,b,c,t;
 scanf("%d%d%d",&a,&b,&c);
 /*以下两个 if 语句使得 a 中存放的数最小*/
 if(a>b){ t=a; a=b; b=t; }
 if(a>c){ t=a; a=c; c=t; }
 /*以下 if 语句使得 b 中存放的数次小*/
```

```
 if(b>c) { t=b; b=c; c=t; }
 printf("%d,%d,%d\n",a,b,c);}
```

2．累加

累加算法的要领是形如“**s=s+**A”的累加式，此式必须出现在循环中才能被反复执行，从而实现累加功能。“A”通常是有规律变化的表达式，s 在进入循环前必须获得合适的初值，通常为 0。

例 1、求 1+2+3+……+100 的和。

```
main()
{int i,s;
 s=0;      i=1;
 while(i<=100)
 {s=s+i;            /*累加式*/
  i=i+1;            /*特殊的累加式*/
 }
 printf("1+2+3+...+100=%d\n",s);}
```

【解析】程序中**加粗**部分为累加式的典型形式，赋值号左右都出现的变量称为累加器，其中“**i = i + 1**”为特殊的累加式，每次累加的值为 1，这样的累加器又称为计数器。

3．累乘

累乘算法的要领是形如“s=s*A”的累乘式，此式必须出现在循环中才能被反复执行，从而实现累乘功能。“A”通常是有规律变化的表达式，s 在进入循环前必须获得合适的初值，通常为 1。

例 1、求 10！

[分析]10！=1×2×3×……×10

```
main()
{int i;   long c;
 c=1;   i=1;
 while(i<=10)
 {c=c*i;          /*累乘式*/
  i=i+1;
 }
 printf("1*2*3*...*10=%ld\n",c);}
```

二、非数值计算常用经典算法

1．穷举

也称为“枚举法”，即将可能出现的每一种情况一一测试，判断是否满足条件，一般采用循环来实现。

例 1、用穷举法输出所有的水仙花数（即这样的三位正整数：其每位数位上的数字的立方和与该数相等，比如：$1^3+5^3+3^3=153$）。

[**法一**]

```
main()
{int x,g,s,b;
 for(x=100;x<=999;x++)
```

```
  {g=x%10;   s=x/10%10;   b=x/100;
   if(b*b*b+s*s*s+g*g*g==x)printf("%d\n",x);}
}
```

【解析】此方法是将 100 到 999 所有的三位正整数一一考察，即将每一个三位正整数的个位数、十位数、百位数一一求出（各数位上的数字的提取算法见下面的“数字处理”），算出三者的立方和，一旦与原数相等就输出。共考虑了 900 个三位正整数。

[法二]

```
main()
{int g,s,b;
 for(b=1;b<=9;b++)
  for(s=0;s<=9;s++)
   for(g=0;g<=9;g++)
    if(b*b*b+s*s*s+g*g*g==b*100+s*10+g)   printf("%d\n",b*100+s*10+g);
}
```

【解析】此方法是用 1 到 9 做百位数字、0 到 9 做十位和个位数字，将组成的三位正整数与每一组的三个数的立方和进行比较，一旦相等就输出。共考虑了 900 个组合（外循环单独执行的次数为 9，两个内循环单独执行的次数分别为 10 次，故 if 语句被执行的次数为 9×10×10=900），即 900 个三位正整数。与**法一**判断的次数一样。

2．排序

（1）冒泡排序（起泡排序）

假设要对含有 n 个数的序列进行升序排列，冒泡排序算法步骤是：

①从存放序列的数组中的第一个元素开始到最后一个元素，依次对相邻两数进行比较，若前者大后者小，则交换两数的位置；

②第①趟结束后，最大数就存放到数组的最后一个元素里了，然后从第一个元素开始到倒数第二个元素，依次对相邻两数进行比较，若前者大后者小，则交换两数的位置；

③重复步骤①n-1 趟，每趟比前一趟少比较一次，即可完成所求。

例 1、任意读入 10 个整数，将其用冒泡法按升序排列后输出。

```
#define n 10
main()
{int a[n],i,j,t;
 for(i=0;i<n;i++)   scanf("%d",&a[i]);
 for(j=1;j<=n-1;j++) /*n 个数处理 n-1 趟*/
  for(i=0;i<=n-1-j;i++)   /*每趟比前一趟少比较一次*/
   if(a[i]>a[i+1]){t=a[i];a[i]=a[i+1];a[i+1]=t;}
 for(i=0;i<n;i++) printf("%d\n",a[i]);}
```

（2）选择法排序

选择法排序是相对好理解的排序算法。假设要对含有 n 个数的序列进行升序排列，算法步骤是：

①从数组存放的 n 个数中找出最小数的下标（算法见下面的“**求最值**”），然后将最小数与第 1 个数交换位置；

②除第 1 个数以外，再从其余 n-1 个数中找出最小数（即 n 个数中的次小数）的下标，将此数与第 2 个数交换位置；

③重复步骤①n-1 趟，即可完成所求。

例 1、任意读入 10 个整数，将其用选择法按升序排列后输出。

```
#define n 10
main()
{int a[n],i,j,k,t;
 for(i=0;i<n;i++) scanf("%d",&a[i]);
 for(i=0;i<n-1;i++)              /*处理 n-1 趟*/
   {k = i;          /*总是假设此趟处理的第一个（即全部数的第 i 个）数最小，k 记录其下标*/
    for(j=i+1;j<n;j++)
       if(a[j] < a[k])   k = j;
    if (k != i){t = a[i]; a[i] = a[k]; a[k] = t;}
   }
 for(i=0;i<n;i++)
  printf("%d\n",a[i]); }
```

（3）插入法排序

要想很好地掌握此算法，先请了解“有序序列的插入算法”，就是将某数据插入到一个有序序列后，该序列仍然有序。插入算法参见下面的“**数组元素的插入**”。

例 1、将任意读入的整数 x 插入一升序数列后，数列仍按升序排列。

```
#define n 10
main()
{ int a[n]={-1,3,6,9,13,22,27,32,49},x,j,k; /*注意留一个空间给待插数*/
   scanf("%d",&x);
   if(x>a[n-2]) a[n-1]=x ; /*比最后一个数还大就往最后一个元素中存放*/
   else     /*查找待插位置*/
   {j=0;
    while( j<=n-2 && x>a[j]) j++;
   /*从最后一个数开始直到待插位置上的数依次后移一位*/
    for(k=n-2; k>=j; k- -)   a[k+1]=a[k];
    a[j]=x; /*插入待插数*/ }
    for(j=0;j<=n-1;j++)   printf("%d   ",a[j]);
}
```

插入法排序的要领就是每读入一个数立即插入到最终存放的数组中，每次插入都使得该数组有序。

例 2、任意读入 10 个整数，将其用插入法按降序排列后输出。

```
#define n 10
main()
{int a[n],i,j,k,x;
 scanf("%d",&a[0]);   /*读入第一个数，直接存到 a[0]中*/
 for(j=1;j<n;j++)     /*将第 2 至第 10 个数一一有序插入到数组 a 中*/
  {scanf("%d",&x);
   if(x<a[j-1]) a[j]=x;   /*比原数列最后一个数还小就往最后一个元素之后存放新读的数*/
```

```
    else     /*以下查找待插位置*/
      {i=0;
       while(x<a[i]&&i<=j-1) i++;
       /*以下 for 循环从原最后一个数开始直到待插位置上的数依次后移一位*/
       for(k=j-1;k>=i;k--) a[k+1]=a[k];
       a[i]=x;       /*插入待插数*/
       }
   }
 for(i=0;i<n;i++)    printf("%d\n",a[i]);
}
```

（4）归并排序

即将两个都**升序（或降序）**排列的数据序列合并成一个仍按原序排列的序列。

例 1、有一个含有 6 个数据的升序序列和一个含有 4 个数据的升序序列，将二者合并成一个含有 10 个数据的升序序列。

```
#define m 6
#define n 4
main()
{int a[m]={-3,6,19,26,68,100} ,b[n]={8,10,12,22};
 int i,j,k,c[m+n];
 i=j=k=0;
 while(i<m && j<n) /*将 a、b 数组中的较小数依次存放到 c 数组中*/
   {if(a[i]<b[j]){c[k]=a[i]; i++;}
    else {c[k]=b[j]; j++;}
    k++; }
 while(i>=m && j<n) /*若 a 中数据全部存放完毕，将 b 中余下的数全部存放到 c 中*/
 {c[k]=b[j]; k++; j++;}
 while(j>=n && i<m) /*若 b 中数据全部存放完毕，将 a 中余下的数全部存放到 c 中*/
 {c[k]=a[i]; k++; i++;}
 for(i=0;i<m+n;i++)    printf("%d    ",c[i]);
}
```

3．查找

（1）顺序查找（即线性查找）

顺序查找的思路是：将待查找的量与数组中的每一个元素进行比较，若有一个元素与之相等则找到；若没有一个元素与之相等则找不到。

例 1、任意读入 10 个数存放到数组 a 中，然后读入待查找数值，存放到 x 中，判断 a 中有无与 x 等值的数。

```
#define N 10
main()
{int a[N],i,x;
 for(i=0;i<N;i++) scanf("%d",&a[i]);
 /*以下读入待查找数值*/
```

```
scanf("%d",&x);
for(i=0;i<N;i++) if(a[i]==x)break ; /*一旦找到就跳出循环*/
if(i<N)    printf("Found!\n");
else    printf("Not found!\n");}
```

（2）折半查找（即二分法）

顺序查找的效率较低，当数据很多时，用二分法查找可以提高效率。使用二分法查找的**前提是数列必须有序**。

二分法查找的思路是：要查找的关键值同数组的中间一个元素比较，若相同则查找成功，结束；否则判别关键值落在数组的哪半部分，就在这半部分中按上述方法继续比较，直到找到或数组中没有这样的元素值为止。

例 1、任意读入一个整数 x，在升序数组 a 中查找是否有与 x 等值的元素。

```
#define n 10
main()
{int a[n]={2,4,7,9,12,25,36,50,77,90};
 int x,high,low,mid;/*x 为关键值*/
 scanf("%d",&x);
 high=n-1;    low=0;    mid=(high+low)/2;
 while(a[mid]!=x&&low<high)
   {if(x<a[mid]) high=mid-1;    /*修改区间上界*/
    else low=mid+1;                /*修改区间下界*/
    mid=(high+low)/2; }
 if(x==a[mid]) printf("Found %d,%d\n",x,mid);
 else printf("Not found\n");
}
```

三、数值计算常用经典算法：

1．级数计算

级数计算的关键是“描述出通项”，而通项的描述法有两种：一为直接法、二为间接法又称递推法。

直接法的要领是：利用项次直接写出通项式；递推法的要领是：利用前一个（或多个）通项写出后一个通项。

可以用直接法描述通项的级数计算例子有：

（1）1+2+3+4+5+……

（2）1+1/2+1/3+1/4+1/5+……等等。

可以用间接法描述通项的级数计算例子有：

（1）1+1/2+2/3+3/5+5/8+8/13+……

（2）1+1/2!+1/3!+1/4! +1/5!+……等等。

（1）直接法求通项

例 1、求 1+1/2+1/3+1/4+1/5+……+1/100 的和。

```
main()
{float s; int  i;
 s=0.0;
```

```
 for(i=1;i<=100;i++) s=s+1.0/i ;
 printf("1+1/2+1/3+...+1/100=%f\n",s);
}
```

【解析】程序中加粗部分就是利用项次 i 的倒数直接描述出每一项，并进行累加。**注意**：因为 i 是整数，故分子必须写成 1.0 的形式！

（2）间接法求通项（即递推法）

例 2、计算下列式子前 20 项的和：1+1/2+2/3+3/5+5/8+8/13+……。

[**分析**]此题后项的分子是前项的分母，后项的分母是前项分子分母之和。

```
main()
{float s,fz,fm,t,fz1;   int i;
 s=1;            /*先将第一项的值赋给累加器 s*/
 fz=1;fm=2;
 t=fz/fm;     /*将待加的第二项存入 t 中*/
 for(i=2;i<=20;i++)
 {s=s+t;
  /*以下求下一项的分子分母*/
  fz1=fz;        /*将前项分子值保存到 fz1 中*/
  fz=fm;          /*后项分子等于前项分母*/
  fm=fz1+fm;   /*后项分母等于前项分子、分母之和*/
  t=fz/fm;}
 printf("1+1/2+2/3+...=%f\n",s);
}
```

下面举一个通项的一部分用直接法描述，另一部分用递推法描述的级数计算的例子：

例 3、计算级数 $\sum_{n=0}^{\infty}\frac{n^2+1}{n!}\left(\frac{x}{2}\right)^n$ 的值，当通项的绝对值小于 eps 时计算停止。

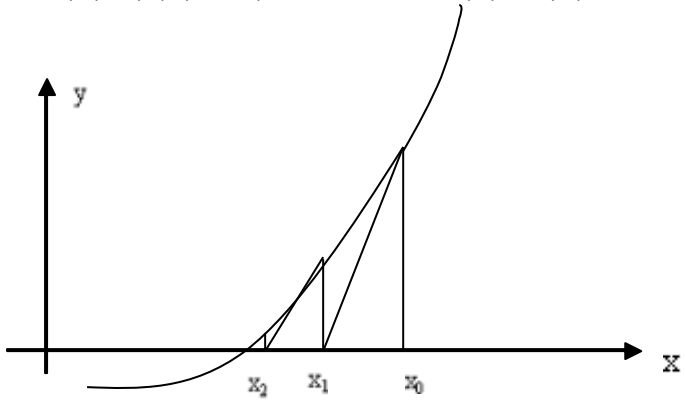
```
#include <math.h>
float g(float x,float eps);
main()
{float x,eps;
 scanf("%f%f",&x,&eps);
 printf("\n%f,%f\n",x,g(x,eps));
}
float g(float x,float eps)
{int n=1;float s,t;
 s=1;   t=1;
 do { t=t*x/(2*n);
      s=s+(n*n+1)*t;   /*加波浪线的部分为直接法描述部分，t 为递推法描述部分*/
      n++; }while(fabs(t)>eps);
 return s;
}
```

2．一元非线性方程求根

（1）牛顿迭代法

牛顿迭代法又称牛顿切线法：先任意设定一个与真实的根接近的值 x_0 作为第一次近似根，由 x_0 求出 $f(x_0)$,过$(x_0，f(x_0))$点做 f(x)的切线，交 x 轴于 x_1，把它作为第二次近似根，再由 x_1 求出 $f(x_1)$, 过$(x_1，f(x_1))$点做 f(x)的切线，交 x 轴于 x_2,……如此继续下去，直到足够接近（比如$|x- x_0|<1e-6$ 时）真正的根 x^*为止。

而 $f'(x_0)=f(x_0)/(x_1- x_0)$　所以 $x_1= x_0- f(x_0)/ f' (x_0)$



例如，用牛顿迭代法求下列方程在 1.5 附近的根：$2x^3-4x^2+3x-6=0$。

```
#include "math.h"
main()
{float x,x0,f,f1;   x=1.5;
 do{x0=x;
     f=2*x0*x0*x0-4*x0*x0+3*x0-6;
     f1=6*x0*x0-8*x0+3;
     x=x0-f/f1; }while(fabs(x-x0)>=1e-5);
 printf ("%f\n",x); }
```

（2）二分法

算法要领是：先指定一个区间$[x_1, x_2]$，如果函数 f(x)在此区间是单调变化的，则可以根据 $f(x_1)$和 $f(x_2)$是否同号来确定方程 f(x)=0 在区间$[x_1, x_2]$内是否有一个实根；如果 $f(x_1)$和 $f(x_2)$同号，则 f(x)在区间$[x_1, x_2]$内无实根，要重新改变 x_1 和 x_2 的值。当确定 f(x) 在区间$[x_1, x_2]$内有一个实根后，可采取二分法将$[x_1, x_2]$一分为二，再判断在哪一个小区间中有实根。如此不断进行下去，直到小区间足够小为止。

具体算法如下：

（1）输入 x_1 和 x_2 的值。

（2）求 $f(x_1)$和 $f(x_2)$。

（3）如果 $f(x_1)$和 $f(x_2)$同号说明在$[x_1, x_2]$ 内无实根，返回步骤（1），重新输入 x_1 和 x_2 的值；若 $f(x_1)$和 $f(x_2)$不同号，则在区间$[x_1, x_2]$内必有一个实根，执行步骤（4）。

（4）求 x_1 和 x_2 的中点：$x_0=（x_1+ x_2）/2$。

（5）求 $f(x_0)$。

（6）判断 $f(x_0)$与 $f(x_1)$是否同号。

①如果同号，则应在$[x_0, x_2]$中寻找根，此时 x_1 已不起作用，用 x_0 代替 x_1，用 $f(x_0)$代替 $f(x_1)$。

②如果不同号，则应在$[x_1, x_0]$中寻找根，此时 x_2 已不起作用，用 x_0 代替 x_2，用 $f(x_0)$代替 $f(x_2)$。

（7）判断 $f(x_0)$的绝对值是否小于某一指定的值（例如 10^{-5}）。若不小于 10^{-5}，则返回步骤（4）重

复执行步骤（4）、（5）、（6）；否则执行步骤（8）。

（8）输出 x_0 的值，它就是所求出的近似根。

例如，用二分法求方程 $2x^3-4x^2+3x-6=0$ 在(-10，10)之间的根。

```
#include "math.h"
main()
{float x1,x2,x0,fx1,fx2,fx0;
 do   {printf("Enter x1&x2");
        scanf("%f%f",&x1,&x2);
        fx1=2*x1*x1*x1-4*x1*x1+3*x1-6;
        fx2=2*x2*x2*x2-4*x2*x2+3*x2-6;
      }while(fx1*fx2>0);
  do {x0=(x1+x2)/2;
      fx0=2*x0*x0*x0-4*x0*x0+3*x0-6;
      if((fx0*fx1)<0)   {x2=x0;   fx2=fx0;}
      else   {x1=x0;   fx1=fx0;}
  }while(fabs(fx0)>1e-5);
  printf("%f\n",x0);}
```

3．梯形法计算定积分

定积分 $\int_a^b f(x)dx$ 的几何意义是求曲线 y=f(x)、x=a、x=b 以及 x 轴所围成的面积。

可以近似地把面积视为若干小的梯形面积之和。例如，把区间[a, b]分成 n 个长度相等的小区间，每个小区间的长度为 h=(b-a)/n，第 i 个小梯形的面积为[f(a+(i-1)・h)+f(a+i・h)]・h/2，将 n 个小梯形面积加起来就得到定积分的近似值：

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} [f(a+(i-1)\bullet h)+f(a+i\bullet h)]\bullet h/2$$

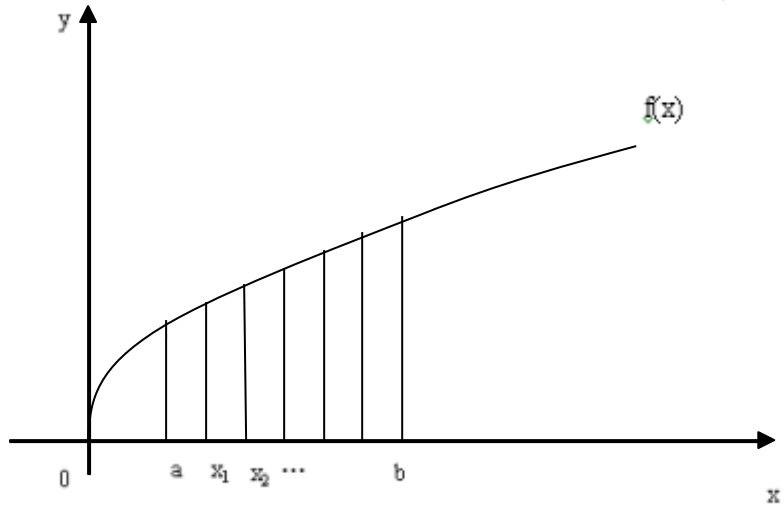
根据以上分析，给出“梯形法”求定积分的 N-S 结构图：

<table>
<tr><td>输入区间端点：a，b</td></tr>
<tr><td>输入等分数 n</td></tr>
<tr><td>h=(b-a)/2,　s=0</td></tr>
<tr><td>i 从 1 到 n
<table><tr><td>si=(f(a+(i-1)*h)+f(a+i*h))*h/2</td></tr><tr><td>s=s+si</td></tr></table></td></tr>
<tr><td>输出 s</td></tr>
</table>

上述程序的几何意义比较明显，容易理解。但是其中存在重复计算，每次循环都要计算小梯形的上、下底。其实，前一个小梯形的下底就是后一个小梯形的上底，完全不必重复计

算。为此做出如下改进：

$$\int_a^b f(x)\,dx \approx h \bullet [f(a)/2 + f(b)/2 + \sum_{i=1}^{n-1} f(a + i \bullet h)]$$



矩形法求定积分则更简单，就是将等分出来的图形当作矩形，而不是梯形。

例如：求定积分 $\int_0^4 (x * x + 3 * x + 2)\,dx$ 的值。等分数 n=1000。

```
#include "math.h"
float DJF(float a,float b)
{float t,h;   int n,i;
 float HSZ(float x);
 n=1000;   h=fabs(a-b)/n;
 t=(HSZ(a)+HSZ(b))/2;
 for(i=1;i<=n-1;i++)   t=t+HSZ(a+i*h);
 t=t*h;
 return(t);
}
float HSZ(float x)
{return(x*x+3*x+2); }
main()
{float y;
 y=DJF(0,4);
 printf("%f\n",y);}
```

四、其他常见算法

1．迭代法

其基本思想是把一个复杂的计算过程转化为简单过程的多次重复。每次重复都从旧值的基础上递推出新值，并由新值代替旧值。

例如，猴子吃桃问题。猴子第一天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个。第二天早上又将剩下的桃子吃掉一半，又多吃了一个。以后每天早上都吃了前一天剩下的一

半零一个。到第 10 天早上想再吃时，就只剩一个桃子了。编程求第一天共摘多少桃子。

```
main()
{int day,peach;
 peach=1;
 for(day=9;day>=1;day--)  peach=(peach+1)*2;
 printf("The first day:%d\n",peach);}
```

又如，用迭代法求 x=$\sqrt{a}$ 的根。

求平方根的迭代公式是：$x_{n+1}=0.5\times(x_n+a/x_n)$

[算法]

（1）设定一个初值 x_0。

（2）用上述公式求出下一个值 x_1。

（3）再将 x_1 代入上述公式，求出下一个值 x_2。

（4）如此继续下去，直到前后两次求出的 x 值（x_{n+1} 和 x_n）满足以下关系：

$$|x_{n+1}-x_n|<10^{-5}$$

```
#include "math.h"
main()
{float a,x0,x1;
 scanf("%f",&a);
 x0=a/2;   x1=(x0+a/x0)/2;
 do{x0=x1;
    x1=(x0+a/x0)/2;
   }while(fabs(x0-x1)>=1e-5);
 printf("%f\n",x1);
}
```

2．进制转换

（1）十进制数转换为其他进制数

一个十进制正整数 m 转换成 r 进制数的思路是，将 m 不断除以 r 取余数，直到商为 0 时止，以反序输出余数序列即得到结果。

注意，转换得到的不是数值，而是数字字符串或数字串。

例如，任意读入一个十进制正整数，将其转换成二至十六任意进制的字符串。

```
void tran(int m,int r,char str[],int *n)
{char sb[]="0123456789ABCDEF";   int i=0,g;
 do{g=m%r;
    str[i]=sb[g];
    m=m/r;
    i++;
   }while(m!=0);
 *n=i;
}
main()
```

```
{int x,r0;      /*r0 为进制基数*/
 int i,n;       /*n 中存放生成序列的元素个数*/
 char a[50];
 scanf("%d%d",&x,&r0);
 if(x>0&&r0>=2&&r0<=16)
    {tran(x,r0,a,&n);
     for(i=n-1;i>=0;i--) printf("%c",a[i]);
     printf("\n"); }
 else   exit(0);
}
```

（2）其他进制数转换为十进制数

其他进制整数转换为十进制整数的**要领**是：“按权展开”，例如，有二进制数 101011，则其十进制形式为 $1\times2^5+0\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0=43$。若 r 进制数 $a_n\ldots\ldots a_2a_1$（n 位数）转换成十进制数，方法是 $a_n\times r^{n-1}+\ldots\ldots a_2\times r^1+a_1\times r^0$。

注意：其他进制数只能以字符串形式输入。

例 1、任意读入一个二至十六进制数（字符串），转换成十进制数后输出。

```
#include "string.h"
#include "ctype.h"
main()
{char x[20];   int r,d;
 gets(x);              /*输入一个 r 进制整数序列*/
 scanf("%d",&r);   /*输入待处理的进制基数 2-16*/
 d=Tran(x,r);
 printf("%s=%d\n",x,d);
}
int Tran(char *p,int r)
{int d,i,cr;   char fh,c;
 d=0;   fh=*p;
 if(fh=='-')p++;
 for(i=0;i<strlen(p);i++)
   {c=*(p+i);
    if(toupper(c)>='A')   cr=toupper(c)-'A'+10;
    else   cr=c-'0';
    d=d*r+cr;
    }
  if(fh=='−') d=-d;
  return(d);
}
```

3．矩阵转置

矩阵转置的**算法要领**是：将一个 m 行 n 列矩阵（即 m×n 矩阵）的每一行转置成另一个 n×m 矩阵的相应列。

例 1、将以下 2×3 矩阵转置后输出。

即将　　1　2　3　　转置成　　1　4

　　4　5　6　　2　5

　　3　6

```
main()
{int a[2][3],b[3][2],i,j,k=1;
 for(i=0;i<2;i++)
    for(j=0;j<3;j++)
       a[i][j]=k++;
 /*以下将 a 的每一行转存到 b 的每一列*/
 for(i=0;i<2;i++)
    for(j=0;j<3;j++)
       b[j][i]=a[i][j];
 for(i=0;i<3;i++)        /*输出矩阵 b*/
    {for(j=0;j<2;j++)
        printf("%3d",b[i][j]);
     printf("\n"); }
}
```

4．字符处理

（1）字符统计：对字符串中各种字符出现的次数的统计。

典型例题：任意读入一个只含小写字母的字符串，统计其中每个字母的个数。

```
#include  ″stdio.h″
main()
{char a[100]; int n[26]={0}; int i; /*定义 26 个计数器并置初值 0*/
 gets(a);
 for(i=0;a[i]!= '\0' ;i++)   /*n[0]中存放'a'的个数，n[1] 中存放'b'的个数……*/
    n[a[i]-'a' ]++;   /*各字符的 ASCII 码值减去'a'的 ASCII 码值，正好得到对应计数器下标*/
 for(i=0;i<26;i++)
  if(n[i]!=0) printf(″%c :%d\n ″, i+'a', n[i]);
  }
```

（2）字符加密

例如、对任意一个只含有英文字母的字符串，将每一个字母用其后的第三个字母替代后输出（字母 X 后的第三个字母为 A，字母 Y 后的第三个字母为 B，字母 Z 后的第三个字母为 C。）

```
#include "stdio.h"
#include "string.h"
main()
{char a[80]= ″China″; int i;
 for(i=0; i<strlen(a); i++)
  if(a[i]>='x'&&a[i]<='z'||a[i]>='X'&&a[i]<='Z')  a[i]= a[i]-26+3;
  else   a[i]= a[i]+3;
 puts(a);}
```

5．整数各数位上数字的获取

算法核心是利用“任何正整数整除 10 的余数即得该数个位上的数字”的特点，用循环从低位到高位依次取出整数的每一数位上的数字。

例 1、任意读入一个 5 位整数，输出其符号位及从高位到低位上的数字。

```
main()
{long x;       int w,q,b,s,g;
 scanf("%ld",&x);
 if(x<0) {printf("-,"); x=-x;}
 w=x/10000;          /*求万位上的数字*/
 q=x/1000%10;        /*求千位上的数字*/
 b=x/100%10;         /*求百位上的数字*/
 s=x/10%10;          /*求十位上的数字*/
 g=x%10;             /*求个位上的数字*/
 printf("%d,%d,%d,%d,%d\n",w,q,b,s,g); }
```

例 2、任意读入一个整数，依次输出其符号位及从**低**位到**高**位上的数字。

[分析]此题读入的整数不知道是几位数，但可以用以下示例的方法完成此题：

例如读入的整数为 3796，存放在 x 中，执行 x%10 后得余数为 6 并输出；将 x/10 得 379 后赋值给 x。再执行 x%10 后得余数为 9 并输出；将 x/10 得 37 后赋值给 x……直到商 x 为 0 时终止。

```
main()
{long x;     scanf("%ld",&x);
 if(x<0) {printf("-    "); x=-x;}
 do                        /*为了能正确处理 0，要用 do_while 循环*/
   {printf("%d    ", x%10);
    x=x/10;
   }while(x!=0);
 printf("\n");
}
```

例 3、任意读入一个整数，依次输出其符号位及从**高**位到**低**位上的数字。

[分析]此题必须借助数组将依次求得的低位到高位的数字保存后，再逆序输出。

```
main()
{long x; int a[20],i,j;
 scanf("%ld",&x);
 if(x<0) {printf("-    "); x=-x;}
 i=0;
 do {a[i]=x%10;
      x=x/10;   i++;
   }while(x!=0);
 for(j=i-1;j>=0;j--)
   printf("%d    ",a[j]);
 printf("\n");
}
```

6．辗转相除法求两个正整数的最大公约数

该**算法的要领**是：假设两个正整数为 a 和 b，先求出前者除以后者的余数，存放到变量 r 中，若 r 不为 0，则将 b 的值得赋给 a，将 r 的值得赋给 b；再求出 a 除以 b 的余数，仍然存放到变量 r 中……如此反复，直至 r 为 0 时终止，此时 b 中存放的即为原来两数的最大公约数。

例 1、任意读入两个正整数，求出它们的最大公约数。

[法一：用 while 循环时，最大公约数存放于 **b** 中]

```
main()
{int a,b,r;
 do   scanf("%d%d",&a,&b);
 while(a<=0||b<=0);  /*确保 a 和 b 为正整数*/
 r=a%b;
 while(r!=0)
   {a=b;b=r;r=a%b;}
 printf("%d\n",b);
}
```

[法二：用 do…while 循环时，最大公约数存放于 **a** 中]

```
main()
{int a,b,r;
 do   scanf("%d%d",&a,&b);
 while(a<=0||b<=0);  /*确保 a 和 b 为正整数*/
 do {r=a%b;a=b;b=r;
    }while(r!=0);
 printf("%d\n",a);
}
```

【引申】可以利用最大公约数求最小公倍数。**提示**：两个正整数 a 和 b 的最小公倍数=a×b/最大公约数。

例 2、任意读入两个正整数，求出它们的最小公倍数。

[法一：利用最大公约数求最小公倍数]

```
main()
{int a,b,r,x,y;
 do   scanf("%d%d",&a,&b);
 while(a<=0||b<=0);  /*确保 a 和 b 为正整数*/
 x=a; y=b;                  /*保留 a、b 原来的值*/
 r=a%b;
 while(r!=0) {a=b;b=r;r=a%b;}
 printf("%d\n",x*y/b);
}
```

[法二：若其中一数的最小倍数也是另一数的倍数，该最小倍数即为所求]

```
main()
{int a,b,r,i;
 do   scanf("%d%d",&a,&b);
 while(a<=0||b<=0);  /*确保 a 和 b 为正整数*/
```

```
 i=1;
 while(a*i%b!=0) i++;
 printf("%d\n",i*a);
}
```

7．求最值

即求若干数据中的最大值（或最小值）。**算法要领**是：首先将若干数据存放于数组中，通常假设第一个元素即为最大值（或最小值），赋值给最终存放最大值（或最小值）的 max（或 min）变量中，然后将该量 max（或 min）的值与数组其余每一个元素进行比较，一旦比该量还大（或小），则将此元素的值赋给 max（或 min）……所有数如此比较完毕，即可求得最大值（或最小值）。

例 1、任意读入 10 个数，输出其中的最大值与最小值。

```
#define N 10
main()
{int a[N],i,max,min;
 for(i=0;i<N;i++) scanf("%d",&a[i]);
 max=min=a[0];
 for(i=1;i<N;i++)
  if(a[i]>max)   max=a[i];
  else   if(a[i]<min) min=a[i];
 printf("max=%d,min=%d\n",max,min);
}
```

8．判断素数

素数又称质数，即“只能被 1 和自身整除的大于 1 的自然数”。判断素数的**算法要领**就是依据数学定义，即若该大于 1 的正整数不能被 2 至自身减 1 整除，就是素数。

例 1、任意读入一个正整数，判断其是否为素数。

```
main()
{int x,k;
 do   scanf("%d",&x);
 while(x<=1);    /*确保读入大于 1 的正整数*/
 for(k=2;k<=x-1;k++)
    if(x%k==0)break;    /*一旦能被 2~自身-1 整除，就不可能是素数*/
 if(k==x)   printf("%d is sushu\n",x);
 else   printf("%d is not sushu\n",x);}
```

以上例题可以用以下两种变形来解决（需要使用**辅助判断的逻辑变量**）：

【变形一】将“2~自身－1”的范围缩小至“2~自身的一半”

```
main()
{int x,k,flag;
 do   scanf("%d",&x);   while(x<=1);
 flag=1;   /*先假设 x 就是素数*/
 for(k=2;k<=x/2;k++)
    if(x%k==0){flag=0; break;}/*一旦不可能是素数，即置 flag 为 0*/
```

```
    if(flag==1)   printf("%d is sushu\n",x);
    else   printf("%d is not sushu\n",x); }
```

【变形二】将“2~自身-1”的范围缩小至“2~自身的平方根”

```
  #include "math.h"
  main()
  {int x,k,flag;
   do     scanf("%d",&x);     while(x<=1);
   flag=1;   /*先假设 x 就是素数*/
   for(k=2;k<=(int)sqrt(x);k++)
      if(x%k==0){flag=0; break;}/*一旦不可能是素数，即置 flag 为 0*/
   if(flag==1)   printf("%d is sushu\n",x);
   else   printf("%d is not sushu\n",x); }
```

例2、用**筛选法**求得100以内的所有素数。

算法为：（1）定义一维数组a，其初值为：2，3，……，100；

（2）若a[k]不为0，则将该元素以后的所有a[k]的倍数的数组元素置为0；

（3）a中不为0的元素，均为素数。

```
#include <math.h>
#include <stdio.h>
 main( )
  {int k,j,a[101];
   clrscr();    /*清屏函数*/
   for(k=2;k<101;k++)a[k]=k;
   for(k=2;k<sqrt(101);k++)
      for(j=k+1;j<101;j++)
         if(a[k]!=0&&a[j]!=0)
           if(a[j]%a[k]==0)a[j]=0;
   for(k=2;k<101;k++) if(a[k]!=0)printf("%5d",a[k]);
   }
```

9．数组元素的插入、删除

（1）数组元素的插入

此算法一般是在已经有序的数组中再插入一个数据，使数组中的数列依然有序。**算法要领**是：

假设待插数据为 x，数组 a 中数据为升序序列。

①先将 x 与 a 数组当前最后一个元素进行比较，若比最后一个元素还大，就将 x 放入其后一个元素中；否则进行以下步骤；

②先查找到待插位置。从数组 a 的第 1 个元素开始找到不比 x 小的第一个元素，设其下标为 i ；

③将数组 a 中原最后一个元素至第 i 个元素依次一一后移一位，让出待插数据的位置，即下标为 i 的位置；

④将 x 存放到 a(i)中。

例题参见前面“‘**排序’中插入法排序的例 1**”。

（2）数组元素的删除

此**算法的要领**是：首先要找到（也可能找不到）待删除元素在数组中的位置（即下标），然后将待删元素后的每一个元素向前移动一位，最后将数组元素的个数减 1。

例 1、数组 a 中有若干不同考试分数，任意读入一个分数，若与数组 a 中某一元素值相等，就将该元素删除。

```
#define N 6
main()
{int fs[N]={69,90,85,56,44,80},x;     int i,j,n;
 n=N;
 scanf("%d",&x); /*任意读入一个分数值*/
 /*以下查找待删分数的位置，即元素下标*/
 for(i=0;i<n;i++)
    if(fs[i]==x)break;
 if(i==n) printf("Not found!\n");
 else           /*将待删位置之后的所有元素一一前移*/
    {for(j=i+1;j<n;j++) fs[j-1]=fs[j];
     n=n-1;     /*元素个数减 1*/
    }
 for(i=0;i<n;i++)printf("%d    ",fs[i]);
}
```

10．二维数组的其他典型问题

（1）方阵的特点

行列相等的矩阵又称方阵。其两条对角线中“\”方向的为主对角线，“/”方向的为副对角线。主对角线上各元素的下标特点为：行列值相等；副对角线上各元素的下标特点为：行列值之和都为阶数加 1。

主对角线及其以下部分（行值大于列值）称为下三角。

例 1、输出如下 5 阶方阵。

```
1    2    2    2    2
3    1    2    2    2
3    3    1    2    2
3    3    3    1    2
3    3    3    3    1
```

```
#define N 5
main()
{int a[N][N],i,j;
 for(i=0;i<N;i++)
    for(j=0;j<N;j++)
       if(i==j) a[i][j]=1;
       else if(i<j) a[i][j]=2;
       else a[i][j]=3;
 for(i=0;i<N;i++)
```

```
    {for(j=0;j<N;j++)
       printf("%3d",a[i][j]);
     printf("\n");
    }
}
```

例 2、输出如下 5 阶方阵。

```
 1   2   3   4   5
 2   3   4   5   6
 3   4   5   6   7
 4   5   6   7   8
 5   6   7   8   9
```

```
#define N 5
main()
{int a[N][N],i,j;
 for(i=0;i<N;i++)
    for(j=0;j<N;j++)
       a[i][j]=i+j+1;      /*沿副对角线平行线方向考察每个元素，其值等于行列值之和+1*/
 for(i=0;i<N;i++)
  {for(j=0;j<N;j++)
     printf("%3d",a[i][j]);
    printf("\n");}
}
```

（2）杨辉三角形

杨辉三角形的每一行是$(x+y)^n$的展开式各项的系数。例如第一行是$(x+y)^0$，其系数为 1；第二行是$(x+y)^1$，其系数为 1，1；第三行是$(x+y)^2$，其展开式为 $x^2+2xy+y^2$，系数分别为 1，2，1；……直观形式如下：

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
……
```

分析以上形式，可以发现其**规律**：是 n 阶方阵的下三角，第一列和主对角线均为 1，其余各元素是它的上一行、同一列元素与上一行、前一列元素之和。

例 1、编程输出杨辉三角形的前 10 行。

```
#define N 10
main()
{int a[N][N],i,j;
 for(i=0;i<N;i++) a[i][0]=a[i][i]=1;
 for(i=2;i<N;i++)
    for(j=1;j<=i-1;j++)
```

```
        a[i][j]=a[i-1][j-1]+a[i-1][j];
  for(i=0;i<N;i++)
   {for(j=0;j<=i;j++)
      printf("%4d",a[i][j]);
    printf("\n");
   }
}
```

例 2、以等腰三角形的形状输出杨辉三角形的前 5 行。

```
                    1
                 1     1
              1     2     1
           1     3     3     1
        1     4     6     4     1
```

```
#define N 5
main()
{int a[N][N],i,j;
 for(i=0;i<N;i++)
   a[i][0]=a[i][i]=1;
 for(i=0;i<N;i++)
   for(j=1;j<i;j++)
   a[i][j]=a[i-1][j-1]+a[i-1][j];
 for(i=0;i<N;i++)
   {for(j=N-i;j>=0;j--)printf("   ");    /*输出时每行前导空格递减*/
    for(j=0;j<=i;j++)
      printf("%4d",a[i][j]);
    printf("\n");
   }
}
```