# 三、递推法

递推法是利用问题本身所具有的一种递推关系求问题解的一种方法。设要求问题规模为 N 的解，当 N=1 时，解或为已知，或能非常方便地得到解。能采用递推法构造算法的问题有重要的递推性质，即当得到问题规模为 i-1 的解后，由问题的递推性质，能从已求得的规模为 1，2，…，i-1 的一系列解，构造出问题规模为 I 的解。这样，程序可从 i=0 或 i=1 出发，重复地，由已知至 i-1 规模的解，通过递推，获得规模为 i 的解，直至得到规模为 N 的解。

【问题】 阶乘计算

问题描述：编写程序，对给定的 n（n≦100），计算并输出 k 的阶乘 k！（k=1，2，…，n）的全部有效数字。

由于要求的整数可能大大超出一般整数的位数，程序用一维数组存储长整数，存储长整数数组的每个元素只存储长整数的一位数字。如有 m 位成整数 N 用数组 a[ ]存储：

N=a[m]×10m-1+a[m-1]×10m-2+ … +a[2]×101+a[1]×100

并用 a[0]存储长整数 N 的位数 m，即 a[0]=m。按上述约定，数组的每个元素存储 k 的阶乘 k！的一位数字，并从低位到高位依次存于数组的第二个元素、第三个元素……。例如，5！=120，在数组中的存储形式为：

3 0 2 1 ……

首元素 3 表示长整数是一个 3 位数，接着是低位到高位依次是 0、2、1，表示成整数 120。

计算阶乘 k！可采用对已求得的阶乘(k-1)！连续累加 k-1 次后求得。例如，已知 4！=24，计算 5!，可对原来的 24 累加 4 次 24 后得到 120。细节见以下程序。

```
# include
# include
# define MAXN 1000
void pnext(int a[ ],int k)
{ int *b,m=a[0],i,j,r,carry;
  b=(int * ) malloc(sizeof(int)* (m+1));
  for ( i=1;i<=m;i++) b[i]=a[i];
  for ( j=1;j<=k;j++)
  { for ( carry=0,i=1;i<=m;i++)
  { r=(i          a[i]=r%10;
  carry=r/10;
  }
  if (carry) a[++m]=carry;
  }
  free(b);
  a[0]=m;
}

void write(int *a,int k)
{ int i;
  printf(“%4d！=”,k);
```

```
  for (i=a[0];i>0;i--)
  printf(“%d”,a[i]);
printf(“\n\n”);
}

void main()
{ int a[MAXN],n,k;
  printf(“Enter the number n: “);
  scanf(“%d”,&n);
  a[0]=1;
  a[1]=1;
  write(a,1);
  for (k=2;k<=n;k++)
  { pnext(a,k);
  write(a,k);
  getchar();
  }
}
```