# 2.1.3 Applications

1、根据问题，进行分析，建立表结构： 一元多项式的表示

$$P_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$$
$$= \sum_{i=0}^{n} a_i x^i$$

- *n*阶多项式$P_n(x)$有*n*+1项。
  - 系数 $a_0, a_1, a_2, \ldots, a_n$
  - 指数 0, 1, 2, …, *n*。按升幂排列

## 第一种表示方法

**建立多项式系数的表 $P_n=(a_0, a_1, a_2, \ldots, a_n)$，用数组表示多项式的系数，数组下标表示指数。**

| | 0 | 1 | 2 | | degree | maxDegree |
|---|---|---|---|---|---|---|
| coef | $a_0$ | $a_1$ | $a_2$ | …… | $a_n$ | ……… |

↑
$n$

适用于指数连续排列、 “0”系数较少的情况。但对于指数不全的多项式，如$P_{20000}(x) = 3 + 5x^{50} + 14x^{20000}$, 会造成系统空间的巨大浪费。

0 1 2 3 ………50 …….. …..20000

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| coef | 3 | 0 | 0 | 0 | | | 5 | 0 | 0 | | 14 | | | | |

## 第二种表示方法

一般情况下，一元多项式可写成：

$P_n(x)=p_1x^{e1}+p_2x^{e2}+\ldots+p_mx^{em}$

其中：$p_i$是指数为*ei*的项的非零系数，

$0 \leq e1 \leq e2 \leq \ldots \leq em \leq n$

抽象成二元组表示 的表：$((p_1,e1),(p_2,e2), \ldots ,(p_m,em))$

例：$P_{999}(x) = 7x^3 - 2x^{12} -8x^{999}$

表示成: ((7,3),(-2,12),(-8,999))

利用数组进行存储如下：

| | 0 | 1 | 2 | | *i* | | *m* |
|---|---|---|---|---|---|---|---|
| coef | $a_0$ | $a_1$ | $a_2$ | …… | $a_i$ | …… | $a_m$ |
| exp | $e_0$ | $e_1$ | $e_2$ | …… | $e_i$ | …… | $e_m$ |

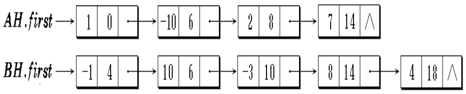
利用链表进行存储如下：

$AH = 1 - 10x^6 + 2x^8 +7x^{14}$

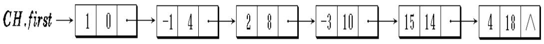
（1,0），（-10,6），（2,8）， （7,14）

$BH = - x^4 + 10x^6 - 3x^{10} + 8x^{14} +4x^{18}$

（-1,4）， （10,6）， （-3,10）， （8,14）， （4,18）



(a) 两个相加的多项式



(b) 相加结果的多项式

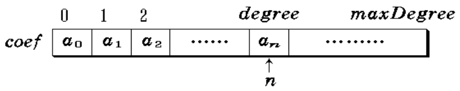
**算法思想：**

```
初始化;
While (两个链都没处理完)
  { if (指针指向当前节点的指数项相同)
        {系数相加，在C链中填加新的节点；
          A、B链的指针均前移；}
     else
        {以指数小的项的系数添入C链中的新节点；
          指数小的相应链指针前移；}
  }
While(A链处理完)
  { 顺序处理B链；}
While(B链处理完)
  { 顺序处理A链；}
```

## 2. 数组与下标的灵活应用

- 规律：当被处理对象之间符合表的特征时建立表，用数组存储表中元素时，对数据元素的加工处理算法，可以充分利用数据元素的下标变化规律。

针对建立的多项式系数的表 $P_n=(a_0, a_1, a_2, \ldots, a_n)$，用数组表示多项式的系数，数组下标表示指数。



每一个问题中的信息往往是多方面的，在算法中一般有输入信息、输出信息和信息加工处理过程中的中间信息。如何确定用数组进行信息存储，数组元素下标与信息如何对应等问题，很大程度上影响着算法的编写效率和运行效率。

下面的例子恰当地选择了用数组存储的信息，并把题目中的有关信息作为下标使用，使算法的实现过程大大简化。

【例1】某次选举，要从五个候选人（编号分别为1、2、3、4、5）中选一名厂长。请编程完成统计选票的工作。

### 算法设计：

1）虽然选票发放的数量一般是已知的，但收回的数量通常是无法预知的，所以算法采用随机循环，设计停止标志为“-1”。

2）统计过程的一般方法为：先为五个候选人各自设置五个“计数器”S1，S2，S3，S4，S5，然后根据录入数据通过多分支语句或嵌套条件语句决定为某个“计数器”累加1，这样做效率太低。

现在把五个“计数器”用数组代替，让选票中候选人的编号xp做下标，执行A[xp]=A[xp]+1就可方便地将选票内容累加到相应的“计数器”中。也就是说数组结构是存储统计结果的，而其下标正是原始信息。

3）考虑到算法的健壮性，要排除对1-5之外的数据进行统计。

## 算 法 描 述：

```
vote( )
{ int i,xp,a[6];
  print(“input data until input -1”);
  input(xp );
  while(xp!=-1)
   { if (xp>=1 and xp<=5 )
         a[xp]=a[xp]+1;
     else
         print(xp, "input error!");
     input(xp );
   }
   for (i=1;i<=5;i++)
      print(i,"number get", a[i], "votes");
}
```

【例2】编程统计身高（单位为厘米）。统计分150-154；155-159；160-164；165-169；170-174；175-179及低于150、高于179共八档次进行。

### 算法设计：

输入的身高可能在50-250之间，若用输入的身高数据直接作为数组下标进行统计，要开辟200多个空间，计数之后还要再次求和。

而统计是分八个档次进行的，统计区间的大小是都固定为5，这样用“身高/5-29”做下标，则只需开辟8个元素的数组，对应八个统计档次，即可完成统计工作。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 150, | 150-154, | 155-159, | 160-164, | 165-169, | 170-174, | 175-179, | 179 |

152/5-29=1　161/5-29=3　170/5-29=5

## 算法如下：

```
main(  )
{ int i,sg,a[8];
print(“input height data until input -1”);
input(sg  );
while   (sg<>-1)
    {if  (sg>179)    a[7]=a[7]+1;
     else
        if  (sg<150)   a[0]=a[0]+1;
        else    a[sg/5-29]=a[sg/5-29]+1;
         input( sg  );
     }
 for (i=0;i<=7;i=i+1)
   print(i+1，“field the number of people：  ”，a[i]);
}
```

【例3】某一次考试共考了语文、代数和外语三科。一个小组共有九人，考后各科及格名单如下表，请编写算法找出三科全及格的学生的名单（学号）。

| 科目 | 及格学生学号 |
|---|---|
| 语文 | 1，9，6，8，4，3，7 |
| 代数 | 5，2，9，1，3，7 |
| 外语 | 8，1，6，7，3，5，4，9 |

## 方 法 一:

- 从语文及格名单中逐一抽出各及格学生学号，先在代数及格名单核对，若有该学号（说明代数也及格了），再在外语及格名单中继续查找，看该学号是否也在外语及格名单中。若仍在，说明该号属全及格学生的学号，否则就是至少有一科不及格的。若语文及格名单中就没有某生的号，显然该生根本不在比较之列，自然不属全及格学生。
- 方法采用了枚举尝试的方法
- A，B，C三组分别存放语文、代数、外语及格名单，尝试范围为三重循环：
- I循环， 初值0， 终值6， 步长1
- J循环， 初值0， 终值5， 步长1
- K循环， 初值0， 终值7， 步长1
- 定解条件： A[I]=B[J]=C[K]
- 共尝试7*6*8=336次。

```
main( )
 {int   a[7], b[6], c[8],i,j,k,v,flag;
  for( i =0;i<=6;i=i+1)    input(a[i]);
  for( i =0;i<=5;i=i+1)    input(b[i]);
  for( i =0;i<=7;i=i+1)    input(c[i]);
  for( i =0;i<=6;i=i+1)
  {v=a[i];
   for( j =0;j<=5;j=j+1)
        if ( b[j]=v )
            for(k =0;k<=7;k=k+1)
                if(c[k]=v)
          {print(v); break;}
   }
}
```

## 方法二：

- 用数组A的九个下标分量作为各学号考生及格科目的计数器。将三科及格名单共存一个数组，当扫描完毕总及格名单后，凡下标计数器的值为3的就是全及格的，其余则至少有一科不及格的。
- 该方法同样也采用了枚举尝试的方法。

- 当题目中的数据缺乏规律时，很难把重复的工作抽象成循环不变式来完成，但先用数组结构存储这些信息后，问题就迎刃而解。

【例4】编程将编号“翻译”成英文。例35706“翻译”成three-five-seven-zero-six。

- 算法设计：
- 1） 编号一般位数较多，可按长整型输入和存储。
- 2） 通过求余、取整运算，可以取到编号的各个位数字。用这个数字通过if **语句**就可以**找到**对应的英文数字。

取一位的操作： **num2 mod 10;**
修改要处理的数：**num2=num2/10**

【例4】编程将编号“翻译”成英文。例35706“翻译”成three-five-seven-zero-six。

- 改进的算法设计：
- 1） 将英文的“zero-nine”存储在数组中，对应下标为0-9。这样无数值规律可循的单词，通过下标就可以方便存取、访问了。

```
main( )
 {int i,a[10], ind;    long num1,num2;
  char eng[10][6]={“zero”,”one”,”two”,”three ”,” four”,
 ” five”,”six”,”seven”,“eight”,”nine”};
  print(“Input a num”);
  input(num1);    num2=num1;  ind =0;
 while (num2<>0)
{a[ind]=num2 mod 10;  ind= ind +1;  num2=num2/10; }
 print(num1, “English_exp:”, eng[a[ind-1]]);
for( i=ind-2;i>=0;i=i-1)
   print(“-”，  eng[a[i]]);
}
```

## 通过数组记录状态信息

有的问题会限定在现有数据中，每个数据只能被使用一次，怎么样表示一个数据“使用过”还是没有“使用过”？

一个朴素的想法是：用数组存储已使用过的数据，然后每处理一个新数据就与前面的数据逐一比较看是否重复。这样做，当数据量大时，判断工作的效率就会越来越低。

【例5】求$x$，使$x^2$为一个各位数字互不相同的九位数。

- 算法分析： 只能用枚举法尝试完成此题。由$x^2$为一个九位数，估算$x$应在10000-32000之间。
- 算法设计：
- 1） 用一个10 个元素的状态数组$p$，记录数字0-9在$x^2$中出现的情况。数组元素都赋初值为1，表示数字0-9没有被使用过。
- 2） 对尝试的每一个数$x$，求$x*x$，并取其各个位的数字，数字作为数组的下标，若对应元素为1，则该数字第一次出现，将对应的元素赋为0，表示该数字已出现一次。否则，若对应元素为0，则说明有重复数字，结束这次尝试。

```
main(  )
{long  x, y1, y2;  int  p[10], 2,  i,  t,  k,  num=0;
 for  (x=10000;x<32000; x=x+1)
   { for(i=0;  i<=9;  i=i+1)            p[i]=1;
     y1=x*x ;  y2=y1;        k=0;
     for(i=1;  i<=9;  i=i+1)
       {t=y2 mod 10;
        y2=y2/10;
        if(p[t]=1)   {k=k+1;    p[t]=0;}
         else         break;
       }
    if(k=9)
       {num=num+1;
         print (“No.”,num , “：  n=”, x, “n²=”, y1);}
     }
}
```

## 3. 优化算法的数学模型

数学建模就是把现实世界中的实际问题加以提炼，抽象为数学模型，求出模型的解，验证模型的合理性，并用该数学模型所提供的解答来解释现实问题，我们把数学知识的这一应用过程称为数学建模。

**【例6】已知有五个数，求前四个数与第五个数分别相乘后的最大数。**

```
max1(int a,b,c,d,e)
{  int x ;
   a=a*e;
   b=b*e;
   c=c*e;
   d=d*e;
   if( a>b)
      x=a;
   else
      x=b;
   if (c>x)
      x=c;
   if (d>x)
      x=d;
   print(x);
   }
```

```
    max2(int a,b,c,d,e)
{ int x
   if (a>b)
     x=a;
   else
     x=b;
    if (c>x)
   x=c;
    if (d>x)
     x=d;
     x=x*e;
   print(x);
   }
```

| 操作<br>算法 | 乘法 | 赋值 | 条件判断 |
|---|---|---|---|
| Max1 | 4 | 7 | 3 |
| Max2 | 1 | 4 | 3 |

**以上两个算法基于的数学模型是不同的，一个算法先积再求最大值，另一个算法先求最大值再求积，求从上表可以看出，后一个算法的效率明显要高于前一个算法。**

- 数学建模的基本方法

从分析问题的几种简单的、特殊的情况中，发现一般规律或作出某种猜想，从而找到解决问题的途径。这种研究问题方法叫做归纳法。即归纳法是从简单到复杂，由个别到一般的一种研究方法。

**【例7】**求$n$次二项式各项的系数。已知二项式的展 开式为:

$$(a+b)^n=C_n^0a^n+C_n^1a^{n-1}b+C_n^2a^{n-2}b^2+\cdots+C_n^nb^n$$

问题分析：若只用的数学组合数学的知识，直接建模

$$C_n^k=\frac{n!}{k!\cdot(n-k)!}$$

$k$=0，1，2，3……$n$。

用这个公式去计算，$n$+1个系数，即使你考虑到了前后系数之间的数值关系：算法中也要有大量的乘法和除法运算，效率低。

数学知识是各阶多项式的系数呈杨辉三角形的规律

$(a+b)^0$ 1
$(a+b)^1$ 1 1
$(a+b)^2$ 1 2 1
$(a+b)^3$ 1 3 3 1
$(a+b)^4$ 1 4 6 4 1
$(a+b)^5$ ……

则求$n$次二项式的系数的数学模型就是求$n$阶杨辉三角形。

算法设计要点： 除了首尾两项系数为1外，当$n$>1时，$(a+b)^n$ *的中间各项系数是*$(a+b)^{n-1}$的相应两项系数之和，如果把$(a+b)^n$的$n$+1的系数列为数组$c$，*则除了*$c(1)$、$c(n+1)$恒为1外，设$(a+b)^n$的系数为$c(i)$，$(a+b)^{n-1}$ 的系数设为$c'(i)$。则有：

$c(i)=c'(i)+c'(i-1)$

而当$n$=1时，只有两个系数 $c(1)$ 和 $c(2)$ (值都为1)。不难看出，对任何$n$, $(a+b)^n$的二项式系数可由$(a+b)^{n-1}$的系数求得，直到$n$=1时，两个系数有确定值，故可写成递归算法。

## 【例8】编程完成下面给“余”猜数的游戏：

你心里先想好一个1~100之间的整数x，将它分别除以3、5和7并得到三个余数。你把这三个余数告诉计算机，计算机能马上猜出你心中的这个数。

游戏过程如下：

```
please think of a number between 1 and 100
your number divided by 3 has a remainder of? 1
your number divided by 5 has a remainder of? 0
your number divided by 7 has a remainder of? 5
let me think a moment…
your number was 40
```

---

问题分析：算法的关键是：找出余数与求解数之间的关系，也就是建立问题的数学模型。

有以下数学模型：

1）当$s=u+3*v+3*w$时，$s$除以3的余数与$u$除以3的余数是一样的。

2）对$s=cu+3*v+3*w$，当$c$除以3余数为1的数时, $s$除以3的余数与$u$除以3的余数也是一样的。证明如下：

$c$ 除以 3余数为1，记$c=3*k+1$，则$s=u+3*k*u+3*v+3*w$，由1)的结论，上述结论正确。

记$a$，$b$，$c$分别为所猜数据$d$ 除以3，5，7后的余数，则 $d=70*a+21*b+15*c$。

为问题的数学模型，其中70称为$a$的系数，21称为$b$的系数,15称为$c$的系数。

---

由以上数学常识，$a$、$b$、$c$的系数必须满足：

1） $b$、$c$的系数能被3整除，且$a$的系数被3整除余1；这样$d$除以3的余数与$a$相同。

2） $a$、$c$的系数能被5整除，且$b$的系数被5整除余1；这样$d$除以5的余数与$b$相同。

3) $a$、$b$的系数能被7整除，且$c$的系数被7整除余1；这样$d$除以7的余数与$c$相同。

由此可见：

$c$的系数是3和5的最公倍数且被7整除余1，正好是15；

$a$的系数是7和5的最公倍数且被3整除余1，最小只能是70；

$b$的系数是7和3的最公倍数且被5整除余1，正好是21。

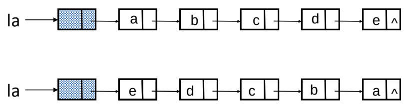
---

算法设计：用以上模型求解的数$d$，可能比100大，这时只要减去3，5，7的最小公倍数就是问题的解了。

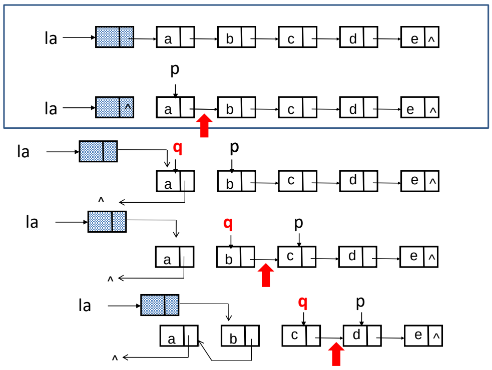
```
main( )
{  int a,b,c,d;
   print( “please think of a number between 1 and 100.”);
   print( “your number divided by 3 has a remainker of”);
   input(a);
   print( “your number divided by 5 has a remainker of”);
   input(b);
   print( “your number divided by 7 has a remainker of”);
   input(c);
   print( “let me think a moment…”);
   for (i=1 ,i<1500; i=i+1);     //消耗时间，让做题者思考
     d=70*a+21*b+15*c;
   while (d>105)       d=d-105;
   print( “your number was ”, d);
 }
```

---

### 4. 单链表的就地逆转



---



# 小 结

- 表的逻辑结构及其特点
- 表的物理结构及其特点
- 表的基本操作算法及性能分析
- 应用实例