第一章　简单程序

无论做任何事情，都要有一定的方式方法与处理步骤。计算机程序设计比日常生活中的事务处理更具有严谨性、规范性、可行性。为了使计算机有效地解决某些问题，须将处理步骤编排好，用计算机语言组成“序列”，让计算机自动识别并执行这个用计算机语言组成的“序列”，完成预定的任务。将处理问题的步骤编排好，用计算机语言组成序列，也就是常说的编写程序。在 Pascal 语言中，执行每条语句都是由计算机完成相应的操作。编写 Pascal 程序，是利用 Pascal 语句的功能来实现和达到预定的处理要求。“千里之行，始于足下”，我们从简单程序学起，逐步了解和掌握怎样编写程序。

第一节　Pascal 程序结构和基本语句

在未系统学习 Pascal 语言之前，暂且绕过那些繁琐的语法规则细节，通过下面的简单例题，可以速成掌握 Pascal 程序的基本组成和基本语句的用法，让初学者直接模仿学习编简单程序。

[例 1.1]编程在屏幕上显示“Hello World!”。

Pascal 程序：

```
Program ex11;
Begin
  Writeln (‘Hello World!’);
  Readln;
End.
```

这个简单样例程序，希望大家的程序设计学习能有一个良好的开端。程序中的 Writeln 是一个输出语句，它能命令计算机在屏幕上输出相应的内容，而紧跟 Writeln 语句后是一对圆括号，其中用单引号引起的部分将被原原本本地显示出来。

[例 1.2]已知一辆自行车的售价是 300 元，请编程计算 a 辆自行车的总价是多少？

解：若总售价用 m 来表示，则这个问题可分为以下几步处理：

①从键盘输入自行车的数目 a；

②用公式　m=300*a　计算总售价；

③输出计算结果。

Pascal 程序：

```
Program Ex12;                    {程序首部}
Var a,m : integer;               {说明部分}
Begin                                {语句部分}
  Write(‘a=’);
```

```
    ReadLn(a);                          {输入自行车数目}
    M := 300*a;                         {计算总售价}
    Writeln('M=',m);                    {输出总售价}
    ReadLn;                             {等待输入回车键}
End.
```

此题程序结构完整，从中可看出一个 Pascal 程序由三部分组成:

(1)程序首部

由保留字 Program 开头，后面跟一个程序名(如:Exl1)；其格式为:

Program　程序名；

程序名由用户自己取，它的第一个字符必须是英文字母，其后的字符只能是字母或数字和下划线组成，程序名中不能出现运算符、标点符和空格。

(2)说明部分

程序中所用的常量、变量，或类型、及过程与自定义函数，需在使用之前预先说明，定义数据的属性（类型）。[例 1.2] 程序中 Var S，R，C: Real；　是变量说明，此处说明 S, R，C 三个变量均为实数类型变量。只有被说明为某一类型的变量，在程序中才能将与该变量同类型的数值赋给该变量。变量说明的格式为:

Var　变量表：类型；

(3)语句部分

指由保留字 Begin (开始)至 End. (结尾)之间的语句系列，是解决问题的具体处理步骤，也是程序的执行部分。

Pascal 程序不管是哪部分，每句末尾都必须有分号(；)，但允许最接近 End 的那个语句末尾的分号省略；程序结束的 End 末尾必须有圆点(.)，是整个程序的结束标志。

程序中花括号“{　}”之间的部分为注释部分。

Pascal 程序结构可归纳用如下的示意图来表示：

Program　程序名；	程序首部
标号说明；　(Label) 常量说明；　(Const) 类型说明；　(Type) 变量说明；　(Var) 过程或函数说明；	说明部分
Begin 语句系列； End.	程序体 (主程序) 语句部分

图 1.1　Pascal 程序的结构

把处理问题的步骤编成能从上到下顺序执行的程序，是简单程序的基本特征。再来分析下面两道例题的 Pascal 程序结构和继续学习基本语句。

［例 1.3］编程计算半径为 R 的圆的面积和周长。

解：这是一个简单问题，按数学方法可分以下几步进行处理：

① 从键盘输入半径的值 R；　　{　要求告诉圆的半径 R }

② 用公式　S= π R²　计算圆面积；

③ 用公式　C=2 π R　计算圆周长；

④ 输出计算结果。

Pascal 程序:

```
Program　Ex13；                    {程序首部 }
Var　R，S，C: Real；               {说明部分 }
Begin                              {语句部分 }
  Write ('R=?');
  Readln(R)；                      {输入半径 }
  S:=Pi*R*R；                      {圆面积公式 S= π R²}
  C:=2*Pi*R；                      {圆周长公式 C=2 π R}
  Writeln('S='，S)；               {输出结果 }
  Writeln('C='，C)；
  Readln                           {等待输入回车键}
End.
```

程序中 Pi 是 Pascal 提供的标准函数，它返回圆周率的近似值：3.1415926…。

(:=)是赋值符号，赋值语句的格式为:

变量:=表达式；

赋值语句的作用是将:=右边表达式的值记录到左边的变量中。

Writeln 是输出语句，输出语句有三种格式:

① Write (输出项 1，输出项 2) ；{执行输出后光标不换行}

② Writeln (输出项 1，输出项 2) ；{执行输出后光标换到下一行}

③ Writeln {仅输出空白且光标换到下一行}

Writeln 语句后面的圆括号以内部分均为输出项，可以是多项，各项间用逗号分隔； 对单引号里的内容按照引号内的原样(字符)输出显示。如果输出项是表达式，则只输出表达式的值，而不是表达式本身。

[例 1.4] 输出两个自然数相除的商和余数。

解:设被除数、除数、商和余数，分别为 A，B，C，D，均为变量，且都是整数类型。题中未给出具体的自然数 A、B，可采用键盘输入方式。

① 给出提示，从键盘输入 a, b；

② 显示两数相除的数学形式；

③ 求出 a 除以 b 的商 c；

④ 求出 a 除以 b 的余数 d；

⑤ 紧接等式后面输出显示商和余数。

Pascal 程序:

```
Program Ex14；
  Var a,b,c,d : integer；
  Begin
    Write('INPUT A，B：');          {给出提示信息}
    Readln(a，b)；                  {输入 a，b}
    Writeln；                       {输出一空行}
```

```
    Write(a，'/'，b，'=');              {输出等式之后不换行}
    c:=a div b;                        {整除运算，取商的整数部分}
    d:=a mod b;                        {相除求余运算，取商的余数部分}
    Writeln(C，'…'，d);                {输出后自动换行 }
    Readln                             {等待输入回车键 }
  End.
```

执行本程序中第一个 Write 语句，输出其引号以内的一串提示信息，是给紧接着的输入语句提供明确的提示(要求)，有“一目了然，人机对话”之效果。

Readln 是一个特殊的输入语句，要求输入一个回车(换行)才能往下执行。

Readln 是输入语句，它的一般格式为:

① Read (变量 1，变量 2);
② Readln (变量 1，变量 2);
③ Readln

前两种格式均要从键盘给变量输入数据，输入时，所键入的数据之间以空格为分隔，以回车为输入结束。若多输入了数据(即数据个数超过变量个数)，Read 语句读完数据之后，能让后续的读语句接着读取多下来的数据；而 Readln 语句对本行多输入的数据不能让后续语句接着读取多下来的数据。为了防止多输入的数据影响下一个输入语句读取数据，建议尽量使用 Readln 语句输入数据。第三种格式不需输入数据，只需按入一个回车键。

[例 1.5]自然数的立方可以表示为两个整数的平方之差，比如 $4^3=10^2-6^2$，请输出自然数 1996 的这种表示形式。(这里的 4^3 用自乘三次的形式 4*4*4 表示；10^2 也用自乘二次的形式 10*10 表示)

解:此题没有现成的计算公式能直接利用，但可以自行推出处理方法或构建适当的运算公式，按着构想的处理方案编排出各步骤。

设这个自然数为 N，两个平方数分别为 X，Y， 将问题表示为求 $N^3=X^2—Y^2$

① 先找出 X 的值，仔细观察题中的示例，用数学方法归纳可得出 X=N*(N+1)/2;（构成本题可用的计算公式）

② 再仔细观察，发现 Y 值比 X 小一个 N 值，即 Y=X—N;

③ 输出等式 $N^3=X^2—Y^2$ 或 N*N*N=X*X—Y*Y

Pascal 程序:

```
Program  Ex15;
  Const  N=1996;            {常量说明 }
  Var    X，Y: Longint;     {变量说明，此题计算中的数值较大，用长整型 }
    Begin
      X:=N*(N+1) div 2;     { div 是整除运算 }
      Y:=X-N;
      Writeln(N,'*',N,'*', N,'=', X,'*', X,'—',Y,'*',Y);      { 输出结果 }
      Readln
    End.
```

本程序中 N 是常量，X，Y 是变量，为长整数类型(Longint)； 程序中的 div 是整除运算，其结果只取商的整数部分；

[例 1.6] 求一元二次方程 $x^2+3x+2=0$ 的两个实数根。

解:方程的系数是常量，分别用 a，b，c 表示，可运用数学上现成的求根公式求方程的根，采取如下方法:

① 先求出 $d=b^2-4ac$；（求根公式中需用开方运算的那部分）

② 再用求根公式算出 x1，x2 的值。(x1，x2 = ?)

③ 输出 x1，x2.

Pascal 程序:

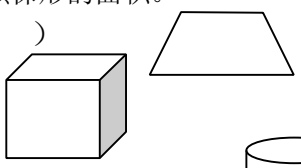
```
program Ex16;
Const a=1;                          {常量说明 }
      b=3;
      c=2;                          {a，b，c 表示方程系数}
Var   d   : integer;                {d 为整型变量}
      X1，X2: Real;                 {X1，X2 为实型变量}
 Begin
   d:=b*b-4*a*c;
   x1:=(-b+sqrt(d))/(2*a);          {求方程的根}
   x2:=(-b-sqrt(d))/(2*a);
   Writeln('X1='，X1，'  ':6，'X2='，X2); {输出结果}
   Readln                           {等待输入一个回车键}
 End.
```

本程序中的 a，b，c 均为常量；变量 d 是整数类型，而变量 x1，x2 则是实数类型，因为运算式中的 Sqrt(d)开平方运算和(/)除法运算使结果为实数。Sqrt( ) 是开平方函数，是 Pascal 系统的一个标准函数。

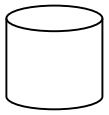
习题 1.1　模仿例题编程

1. 加法计算器：编程由键盘输入两个整数 a 和 b，计算出它们的和并输出到屏幕上。
2. 某梯形的上底、下底和高分别为 8，12，9，求该梯形的面积。

   (梯形面积公式为 $S=\frac{(上底+下底)\times 高}{2}$)



3. 求右图所示边长为 5.6　的正立方体表面积。



4. 已知图园柱体的高为 12，底面园的半径为 7，求园柱体表面积。

5. 计算某次考试语文、数学、英语和计算机等四科的总成绩与平均成绩。
(请用输入语句从键盘输入各科成绩分)

第二节　顺序结构程序与基本数据类型

前面的简单程序已体现出处理问题步骤、思路的顺序关系，这就是顺序结构程序。

［例 1.7］交换两个变量的值：由键盘输入两个正整数 A 和 B，编程交换这两个变量的值。

解：交换两个变量的值，可以想象成交换两盒录音带（称为 A 和 B）的内容，可以按以

下步骤处理：

步骤①：拿一盒空白录音带 C 为过渡，先将 A 翻录至 C；

步骤②：再将 B 翻录至 A；

步骤③：最后将 C 翻录至 B。

这样操作，可达到题目要求。

Pascal 程序：

```
Program Exam17;
Var a,b,c   : integer;
Begin
  Write('A,B=');
  Readln(a,b);
  C:= A;              {等价于步骤 1}
  A := B;             {等价于步骤 2}
  B := C;             {等价于步骤 3}
  Writeln(A,B);
End.
```

[例 1.8] 分钱游戏。甲、乙、丙三人共有 24 元钱，先由甲分钱给乙、丙两人，所分给的数与各人已有数相同；接着由乙分给甲、丙，分法同前；再由丙分钱给甲、乙，分法亦同前。经上述三次分钱之后，每个人的钱数恰好一样多。 求原先各人的钱数分别是多少?

解:设甲、乙、丙三人的钱数分别为 A，B，C。用倒推(逆序)算法， 从最后结果入手，按反相顺序，分步骤推算出每次各人当时的钱数（在每个步骤中，各人钱数分别存在 A、B、C 中）

步骤①: A=8 B=8 C=8 {这是最后结果的钱数，三人都一样多 }

步骤②: A=A/2 (=4) B=B/2 (=4) C=A+B+C(=16) { A，B 未得到丙分给的钱时，只有结果数的一半；C 应包含给 A，B 及本身数三者之和 }

步骤③: A=A/2 (=2) C=C/2 (=8) B=A+B+C(=14) {A，C 未得到乙分给的钱时，只有已有数的一半；B 应包含给 A，C 及本身数三者之和 }

步骤④: B=B/2 (=7) C=C/2 (=4) A=A+B+C(=13)

C 未得到甲分给的钱时，只有已有数的一半；A 应包含给 B，C 及本身数三者之和 }

步骤⑤: 输出 A（=13）B（=7）C（=4）{此时的 A，B，C 就是三人原先的钱数 }

Pascal 程序:

```
Program Exam18；
Var a，b，c: integer；
Begin
  a:=8；  b:=8；  c:=8；                          {对应于步骤①}
  a:=a div 2；  b:=b div 2；  c:=a+b+c；          {对应于步骤②}
  a:=a div 2；  c:=c div 2；  b:=a+b+c；          {对应于步骤③}
  b:=b div 2；  c:=c div 2；  a:=a+b+c；          {对应于步骤④}
  Writeln('a=', a, '  ': 4, 'b=', b, '  ': 4, 'c=', c) ；   {输出}
  Readln
End.
```

细心观察，会发现本程序语句的顺序很关键。此例用反推顺序(逆序)，按步骤正确推算

出各变量的值。当然，有的问题可按正序步骤编程，这类程序都称为顺序程序。

本程序 Writeln 语句的输出项含有(' ': 4)，这里的冒号用来指定该项显示所占宽度，此处是输出 4 个空格即(空格项占 4 格)。

[例 1.9] 有鸡兔同笼，头 30，脚 90，究竟笼中的鸡和兔各有多少只?

解:设鸡为 J 只，兔为 T 只，头为 H，脚为 F，则:

J+T=30 ①

2*J+4*T=90 ②

解此题暂不必采用数学上直接解方程的办法，可采用“假设条件与逻辑推理”的办法：

假设笼中 30 个头全都是兔，那么都按每头 4 只脚计算，总脚数为(4*H)，与实际脚数 (F)之差为(4*H—F)，如果这个差=0，则笼中全是兔（即鸡为 0 只）；如果这个差值 >0，说明多计算了脚数，凡是鸡都多给算了两只脚，用它除以 2 就能得到鸡的只数，处理步骤为:

① J=(4*H—F)/2 {先用脚数差值除以 2 算出鸡的只数}

② T=H—J {再用总头数减鸡数算出兔的只数}

按此方法，这两步运算必须注意先后顺序才会符合运算逻辑。

Pascal 程序:

```
Program Exam16；
Const H=30；                    {常量说明 }
      F=90；
Var J，T: byte；                {为字节类型的整数 }
Begin
  J:=(4*H-F) div 2；        {整除运算 }
  T:=H-J
  Writeln ('J='，J，'  ': 6，'T= '，T ) ；
  Readln
End.
```

本程序中 H，F 为常量，变量 J，T 为 byte 类型，属于整数类型。

Pascal 定义了五个标准整数类型，如下表所示:

类型	取值范围	占字节数	格式
Shortint（短整型）	-128..127	1	带符号 8 位
Integer （整型）	-32768..32767	2	带符号 16 位
Longint（长整型）	-2147483648..2147483647	4	带符号 32 位
Byte （字节型）	0..255	1	无符号 8 位
Word （字型）	0..65535	2	无符号 16 位

在前面程序中常用的数据类型除整数类型,还有实数类型。Pascal 还定义了五个标准实数类型，列表所示如下:

类型	取值范围	占字节数	有效数字
Real	2.9×10^{-39}~1.7×10^{38}	6	7~8 位
Single	1.5×10^{-45}~3.4×10^{38}	4	11~12 位
Double	5.0×10^{-324}~1.7×10^{308}	8	15~16 位
Extended	1.9×10^{-4951}~1.1×104932	10	19~20 位
Comp	$-2^{63}+1$~$2^{38}-1$	8	19~20 位

在 Turbo Pascal 中实数的表示用科学记数法，可认为由三部分组成:

#. ##E+## 或 #.##E-##

① #. ##表示有效数字； ② E 表示以 10 为底的幂； ③ +##或-##是指数部分，+号可省略。

例如: 1.7E+38 可写成 1.7E38 (等同于 1.7×10^{38})。

在实数类型定义下，即使是整数，在程序执行时系统也将自动转换成科学记数形式，试请运行下面程序并注意观察运行结果:

```
Program Exam17;
Var x: real;                    {x 为实数类型 }
Begin
  X:=180;                       {把整数 180 赋给实数类型变量 X}
  Writeln ('x=', x) ;           {输出的 x 自动表示成实数形式 }
  Readln
End.
```

习题 1.2

1.已知△ABC 中的三边长分别为 25.76，74.03，59.31，求△ABC 的面积。

(计算公式: S= 。 其中 P =)

2.某车棚存有自行车和三轮车共 65 辆，它们的轮子数合计为 150 个。求该棚内存有的自行车和三轮车各是多少辆?

3.甲、乙、丙三人分别有磁带 36，48，64 盒。先由甲把自己的磁带平均分为三份，分给乙、丙各一份，自己留下一份；接着是乙，最后是丙，都按甲的方法处理。编程输出甲、乙、丙在上述过程中各人的磁带数分别是多少? (输出所有的中间结果)

4.五位好朋友相聚。第一位朋友带来了很多糖块赠送给各位朋友，使每人的糖块在各自原有的基础上翻了一倍；接着第二位好友也同样向每人赠送糖块，他同样使每人的糖块在各人已有的数量上翻了一倍；第三、第四、第五位好友都照此办理。经过这样的赠送之后，每人的糖块恰好都为 32 块。问各位好友原先的糖块数分别是多少?

第二章　分支程序

在程序设计中，许多问题是在一定条件下才选择某种处理方式的，这就需要用条件判断语句或情况选择语句进行处理。程序执行中将出现选择(分支)，根据条件只选择执行部分语句,不一定都是按原顺序从头到尾地执行所有语句，这样的程序称为分支程序。

第一节　条件语句与复合语句

[例 2.1] 某服装公司为了推销产品，采取这样的批发销售方案:凡订购超过 100 套的，每套定价为 50 元，否则每套价格为 80 元。编程由键盘输入订购套数，输出应付款的金额数。

解:设 X 为订购套数，Y 为付款金额，则:

$$Y=\begin{cases}50*X & (X>100) \quad \{\text{如果 X > 100 就用 Y=50*X 计算}\}\\ 80*X & (X<=100) \quad \{\text{否则用 Y=80*X 计算}\}\end{cases}$$

① 输入 X;

② 判断 X 值；

③ 根据判断结果选择符合条件的那种方法计算 Y 值；

④ 输出计算结果。

Pascal 程序:

```
Program Exam21；
Var x，y: integer；
Begin
  Write('X=') ；Readln(x) ；                    {  输入 X}
  if x >100 then y:=50*X   else   y:＝80*X；    {条件判断与选择 }
  Writeln('y='，y) ；
  Readln
End.
```

程序中的 if 语句常称为条件语句，它的一般格式为:

(1) if 条件 then 语句；

(2) if 条件 then 语句 1 else 语句 2；

IF 语句的功能是按条件在两种可能中选择其中一种。习惯上把 if 后面的表达式称为条件，then 后面的语句称为真项，else 后面的语句称为假项。若条件成立(为真)就执行真项，然后执行 if 语句的后继语句；若条件不成立(为假)就跳过真项而执行假项，然后执行后继语句。而第一种格式只有真项，没有假项，当条件不成立(为假)就什么也不需做，直接往下去执行后继语句。

[例 2.2] 读入三个不同的数，编程按由小到大的顺序排列打印出来。

解:设读入的三个数为 a，b，c，为了把较小的数排在前面，可作如下处理:

① 如果 a＞b 就交换 a、b 的值，将较大的值换至后面；

② 如果 a＞c 就交换 a、c 的值，将较大的值换至后面；

③ 如果 b＞c 就交换 b、c 的值，将较大的值换至后面；

④ 输出处理后的 a,b,c。

Pascal 程序:

```
Programm  Exam22；
Var a，b，c，t: Real；
Begin
    Write('Input a，  b，c=')；
    Readln(a，b，c)；
    if a＞b then
      begin                          {复合语句}
        t:=a；  a:=b；  b:=t         {交换 a，b}
      end；
    if a＞c then
      begin                          {复合语句}
        t:=a；  a:=c；  c:=t          {交换 a，c}
      end；
    if b＞c then
       begin                         {复合语句}
         t:=b；  b:=c；  c:=t            {交换 b，c}
       end；
    Writeln('a，b，c:'，a:6，  b:6，  c:6)；
    Readln
End.
```

if 语句规定它的真项或假项位置上只能是一个基本语句，如果需要写一组语句，就应当使用复合语句。本程序中有三处用到复合语句。每个复合语句的范围是从 Begin 开始到与它相对应的 End 为止。复合语句的地位和一个基本语句相同；其一般格式为：

```
Begin
  语句系列
End；
```

习题 2. 1

1.假设邮局规定寄邮件时若每件重量在 1 公斤以内(含 1 公斤)，按每公斤 1.5 元计算邮费，如果超过 1 公斤时，其超出部分每公斤加收 0.8 元。请编程序计算邮件收费。

2.输入三个正整数，若能用这三个数作为边长组成三角形，就计算并输出该三角形的面积，否则输出 Can't。(组成三角形的条件为：任意两边之和大于第三边)

3.输入一个三位数的整数，将数字位置重新排列，组成一个尽可大的三位数。例如：输入 213，重新排列可得到尽可能大的三位数是 321。

第二节　情况语句与算术标准函数

如果有多种（两种或两种以上）选择，常用情况语句编程。

将前面[例 2.1]改成用如下方法来处理。根据题意，付款计算可分为两种情况：

① Y=50*X　　(X＞100)

② Y=80*X　　(X＜=100)

显然，情况①与②的选择取决于 X 值。假设用 N 表示“情况值”，暂且先让 N＝2；如果 X＞100 则 N=1；（此题中 N 的值只是 1 或 2，且取决于 X 值）

Pascal 程序:

```
Program Exam21_1；
Var X，Y，N: integer；
Begin
  Write('X=') ；readln(x) ；  n:=2；        { 先让 n=2 }
  if X＞100 then n:=1；                     {如果 X＞100 则 n=1 }
  Case n  of                                { 关于情况处理 }
    1: Y:=50*X；
    2: Y:=80*X；
  end；
  Writeln('Y='，Y) ；
  Readln
End.
```

程序中的 Case—end 语句为情况语句，是多路分支控制，一般格式为:

```
Case 表达式 of
  情况常量表 1: 语句 1；
  情况常量表 2: 语句 2；
      :          :
  情况常量表 n: 语句 n
 end；
```

执行情况语句时，先计算 Case 后面表达式的值，然后根据该值在情况常量表中的“对应安排”，选择其对应的语句执行，执行完所选择语句后就结束 Case 语句；如果常量表中没有一个与表达式值对应的语句，则什么也不做就结束本 Case 语句。

Case 语句的另一种应用格式为：

```
Case 表达式 of
  情况常量表 1: 语句 1；
  情况常量表 2: 语句 2；
      :          :
  情况常量表 n: 语句 n；
  else  语句 n+1
end；
```

这种格式的前面部分是相同的，所不同的是：如果常量表中没有一个与表达式值对应的语句，则执行与 else 对应的语句，然后结束 Case 语句。

[例 2.2] 对某产品征收税金，在产值 1 万元以上征收税 5%；在 1 万元以下但在 5000 元以上的征收税 3%；在 5000 元以下但在 1000 元以上征收税 2%；1000 元以下的免收税。编程计算该产品的收税金额。

解:设 x 为产值，tax 为税金，用 P 表示情况常量各值，以题意中每 1000 元为情况分界:

P=0: tax=0 (x<1000)

P=1，2，3，4: tax=x*0.02 (1000<=x<5000)

P=5，6，7，8，9: tax=x*0.03 (5000<X<=10000)

P=10: tax=x*0.05 (x> 10000)

这里的 P 是“情况”值，用产值 x 除以 1000 的整数值作为 P，如果 P>10 也归入 P=10 的情况。Pascal 语言用 P=trunc(x/1000)取整计算，

Pascal 程序:

```
Program Exam22；
Var   x，p :   integer；
      Tax   : real；
Begin
   Write('Number=') ；   readln(x) ；
   P:=trunc(x/1000) ；
   if P＞9 then P:=10；
   Case   P   of
      0              :   tax:=0；
      1,2,3,4        :   tax:=x*0.2；
      5,6,7,8,9      :   tax:=x*0.3；
      10             :   tax:=x*0.5
   end；
   Writeln('tt='，tt:5:2) ；
   Readln
End.
```

情况表达式的计算必须考虑到“全部”情况，不要有遗漏。如果情况常量表的“值”在某范围内是连续的，可将常量表写成：

n1.. n2：语句；

因此，上面程序中的情况常量表可以写成如下程序中表示形式：

```
Program Exam22_1；
Var x,p: integer；
    tax: real；
Begin
   Write('Number=') ；   readln(x) ；
   P:=trunc(x/1000) ；
   if P＞9 then P:=10；
   Case   P   of
     0     : tax:=0；
```

```
    1..4   : tax:=x*0.2;    { 从 1 至 4 作为同一情况处理 }
    5..9   : tax:=x*0.3;    { 从 5 至 9 作为同一情况处理 }
    10     : tax:=x*0.5
  end;
  Writeln('tt=', tt:5:2) ;
  Readln
End.
```

程序中的 trunc(x)为取整函数，是 Pascal 的算术标准函数之一。Pascal 常用的算术标准函数有 19 个:

(1) abs(x) 求 x 的绝对值(|x|);

(2) exp(x) 求 ex 的值; （e 为无理数 2.71828…）

(3) frac(x)求 x 的小数部分;

(4) int(x) 求 x 的整数部分(不舍入，函数值为实型);

(5) ln(x) 求以 e 为底的 x 的对数（log ex ）;

(6) odd(x) 判断 x 的奇偶数(当 x 为奇数时 odd(x)值为 true，否则为 false);

(7) ord(x) 求 x 的序号，结果为整型(x 为有序类型量);

(8) pi π 值(3.1415926535897932…);

(9) pred (x) 求 x(有序类型)的前趋值;

(10) succ(x) 求 x(有序类型)的后继值;

(11) random 随机函数，产生 0～1 的随机值;

(12) random(n)产生 0～n 的随机数(n 为 word 类型，先执行 randomize， 才能得到随机整数);

(13) round(x) 求 x 的四舍五入整数;

(14) trunc(x) 求 x 的整数部分(截掉小数部分，结果为整型);

(15) sqr(x) 求 x 的平方值(x2);

(16) sqrt(x) 求 x 的开平方根值( );

(17) sin(x) 求 x 的正弦函数(x 为弧度);

(18) cox(x) 求 x 的余弦函数(x 为弧度);

(19) arctan(x) 正切的反三角函数(x 为数值);

习题 2.2

1.运输公司计算运费时，距离(S)越长，每公里运费越低，标准如下:

如果 S＜250 公里；运费为标准运价的 100%

如果 250 公里＜＝S＜500 公里，运费为标准运价的 98%;

如果 500 公里＜＝S＜1000 公里，运费为标准运价的 95%;

如果 1000 公里＜＝S＜2000 公里，运费为标准运价的 92%;

如果 2000 公里＜＝S＜3000 公里，运费为标准运价的 90%;

如果 S＝＞3000 公里，运费为标准运价的 85%;。请编计算运费的程序。

2. 输入考试成绩，如果获 85 分以上为 A 等，获 60 分～84 分为 B 等，60 分以下为 C 等，编程输出考试等级。

3. 某车间按工人加工零件的数量发放奖金，奖金分为五个等级：每月加工零件数 N < 100 者奖金为 10 元；100 < = N < 110 者奖金为 30 元；110 < = N <120 者奖金为 50 元；120 < = N

<130 者奖金为 70 元；N > 130 者为 80 元。

请编程，由键盘输入加工零件数量，显示应发奖金数。

第三章　循环程序

在编程中经常遇到需要多次规律相同的重复处理，这就是循环问题。Turbo Pascal 采用不同的循环方式来实现，常用的环循有三种:　for、repeat、while.

第一节　for 循环

for 循环是一种自动计数型循环。

[例 3.1] 试打印出 1~20 的自然数。

解：① 用 a 代表 1~20 各数，同时也用 a 兼作计数，以控制循环次数；

② 让 a 从 1 开始；

③ 输出 a；

④ a 自动计数（加 1），如果未超越所规定的循环范围则重复步骤③，否则结束循环。

Pascal 程序:

```
Program Exam12；
Var a: byte；
Begin
  for a:=1 to 20 do
    Writeln (a)；
  Readln
End.
```

程序中　for a:=1 to 20 do Writeln (a)；　是 for 循环语句。

for 循环语句有两种格式:

(1) for　循环变量:=初值　To 终值 do 语句；

(2) for　循环变量:=初值　downto 终值 do 语句；

第(1)种格式的初值小于等于终值，循环变量值按自动加 1 递增变化；

第(2)种格式的初值大于或等于终值，循环变量值按自动减 1 递减变化。for 循环是 (以递增 1 或以递减 1) 计数型循环。

比如: 若将[例 3.1]程序改为倒计数(递减)循环，则输出 20～1 的自然数数:

```
Program Exam31；
Var a: byte；
Begin
  for a:=20 downto 1 do
    Writeln(a) ；
  Readln
End.
```

[例 3.2]打印出 30 至 60 的偶数。]

解：

方法一：

①设 a 表示 30 至 60 的所有的数，可用 for 循环列出；

②用式子　a mod 2=0　筛选出其中的偶数并输出。

Pascal 程序：

```
Program ex32;
Var a       : integer;
Begin
   For a := 30 to 60 do
      If (a mod 2=0) then writeln(a);
   Readln;
End.
```

在这个程序中，for 循环后的循环语句是一个条件分支语句。

方法二：我们知道，在式子 a=2*n 中，若 n 取自然数 1、2、3、…，时，则 a 依次得到偶数 2、4、6、…。因此要想得到 30 至 60 的偶数，就可以让上面式子中的 n 取 15 至 30 的自然数就可以了。所以本题还可以按以下步骤处理：

①设 n 表示 15 至 30 的所有自然数，可用 for 循环列出；

②用式子　a := 2*n　求出其中的偶数；

③将结果输出至屏幕。

Pascal 程序：

```
Program ex32;
Begin
   For n := 15 to 30 do
      Begin
         a := 2*n;
         Writeln(a);
      End;
   Readln;
End.
```

[例 3.3]自然数求和：编一个程序，求从 1 至 100 的自然数的和。

解：① 令 S＝0；

② 令 a 表示 1 至 100 的自然数，用循环列出；

③ 将这些自然数用公式 S:=S+a 逐一累加到 S 中去；

④ 循环结束后，S 即为 1 至 100 的自然数的和，输出即可。

Pascal 程序：

```
Program ex33;
var s,a : integer;
Begin
   S := 0;
```

```
    For a := 1 to 100 do
      S := S+a;
    Writeln('S=',S);
    Readln;
  End.
```

[例 3.4]一个两位数 x，将它的个位数字与十位数字对调后得到一个新数 y，此时 y 恰好比 x 大 36，请编程求出所有这样的两位数。

解：① 用 for 循环列举出所有的两位数，x 为循环变量；

② 用公式 a:= x div 10 分离出 x 的十位数字；

③ 用公式 b:= x mod 10 分离出 x 的个位数字；

④ 用公式 y:= b*10+a 合成新数 y；

⑤ 用式子 y-x=36 筛选出符合条件的数 x 并输出。

Pascal 程序：

```
Program ex34;
Begin
  For x := 10 to 99 do
  Begin
    a := x div 10;
    b := x mod 10;
    y := b*10+a;
    if y-x=36 then writeln(x);
  End;
  Readln;
End.
```

[例 3.5] 把整数 3025 从中剪开分为 30 和 25 两个数，此时再将这两数之和平方，$(30+25)^2=3025$ 计算结果又等于原数。求所有符合这样条件的四位数。

解:设符合条件的四位数为 N，它应当是一个完全平方数，用(a*a)表示。

① 为了确保 N=(a*a)在四位数（1000～9999）范围内，可确定 a 在 32～99 循环；

② 计算 N=a*a；将四位数 N 拆分为两个数 n1 和 n2；

③ 若满足条件(n1+n2)*(n1+n2)＝N 就输出 N 。

Pascal 程序:

```
Program   Exam35；
Var N，a，  x，n1，n2: Integer；
Begin
  for a:=32 to 99 do
  begin
    N:=a*a；
    n1:= N div 100；          {拆取四位数的前两位数}
    n2:= N-n1*100；           {拆取四位数的后两位数}
    X:=n1+n2；
```

```
    if  x*x=N  then  writeln (N);
  end;
  Readln
End.
```

[例 3.6]用“*”号打印出如下的长方形图案。

```
*********
*********
*********
*********
```

解：① 上面给出的图例共有 4 行，我们可以用一个循环控制行的变化；

② 在每行中又有 9 列，我们可以在前面控制行的循环中再套一个循环来控制列的变化。

Pascal 程序：

```
Program ex36;
Begin
  For a := 1 to 4 do                {外循环控制行的变化}
  Begin
    For b := 1 to 9 do                  {内循环控制列的变化}
      write('*');
    Writeln;                            {输出一行的“*”后换行}
  End;
  Readln;
End.
```

程序中的循环对于 a 的每个值都包含着一个 b=(1～9)次的内循环。外循环 for a 将内循环 for b 包含在里面，称为 for 循环的嵌套。嵌套形式如:

```
for a:=n1 to n2 do
    for b:=m1 to m2 do   循环体语句;
```

[例 3.7] 打印出九九乘法表:

解:设 a 为被乘数，范围为 1～9；b 为乘数，范围为 1～a；乘式为 a*b=(a,b 的乘积)，则

```
a=1:    b=1～a   1*1=1
a=2:    b=1～a   2*1=2   2*2=4
a=3:    b=1～a   3*1=3   3*2=6   3*3=9
a=4:    b=1～a   4*1=4   4*2=8   4*3=13   4*4=16
  :       :
a=9     b=1～a   9*1=9   9*2=18  …        9*9=81
```

⑴从上面分解的横行中看到共有 9 行，这里的“行”数变化与 a 的变化从 1～9 相同，可用 a 控制“行”的循环；

⑵每“行”里面相乘的次数与 b 的范围相关，由 b 控制每“行”里面的“内部”循环；

⑶内循环被包含在最里层，执行完每“行”的内部循环，就到下一“行”去执行新“行”里面的循环，每“行”都拥有形式相同的（ b=1～a ）内循环。

即每到一“行”都要执行该“行”的内循环。这里所指的“行”可以理解成抽象的行，不一定是实际上具体对应的行，可以是一个处理“块”。

Pascal 程序:

```
Program Exam37；
Var a，b: byte；
Begin
  for a:=1 to 9 do                {外循环 }
   begin
     for b:=1 to a do                          {内循环 }
       write(a,’ * ’，b,’ = ’，a*b,’   ’:3)；
     writeln
   end；
  Readln
End.
```

根据这种格式还可以实现多层循环嵌套,例如：

```
            for a:=n1 to n2 do
                for b:=m1 to m2 do
                    for c:=k1 to k2 do   循环体语句；
```

[例 3.8]从七张扑克牌中任取三张，有几种组合方法？请编程输出所有组合形式。

解：设每次取出三张分别为 a,b,c。用三重循环分别从 1～7 的范围里取值；为了排除取到重号，用(a-b)*(b-c)*(a-c) < >0 进行判断。

Pascal 程序：

```
program Exam38;
const   n=7;
var   a,b,c,t: integer;
Begin
  t:=0;
  for a:=1 to n do
    for b:=1 to n do
      for c:=1 to n do
        if (a-b) * (b-c) * (a-c) < >0 then
        Begin
            inc (t);
          writeln (a:3, b:3, c:3)
        End;
      writeln ( total:, t :5);
      readln
  End.
```

[例 3.9] 数学上把除了 1 和它本身，没有别的数能够整除它的自然数叫做素数(或质数)。现在由键盘输入一个自然数 N，编程判断 N 是否是素数，是则输出“Yes”，否则输出“No”。

解：根据定义，对于给定的自然数 N，只需判断除 1 和它本身外，还有没有第三个自然数即可。

① 令 K 从 1 循环至 N；

② 根据 N mod K 是否为 0 可统计 K 的约数的个数；

③ 若 N 的约数的个数超过 2 个，则判定 N 不是素数。

Pascal 程序:

```
Program Exam39；
Var   n，m，k，t: integer；
Begin
  write(‘N=’);
  ReadLn(N);
  t:=0；
  for   k:=1 to N do                          {外循环 }
    if   N mod k=0 then t := t+1;                          {如果 N 是奇数 }
  if t>2 then writeln(‘No’)
       else writeln(‘Yes’);
  Readln;
End.
```

程序中的变量 yse 为布尔(或逻辑)类型(Boolean)。布尔值只有两个：

True(真)　　False(假)

布尔值与条件判断结果为真(条件成立)或为假(条件不成立)的作用相同，常用于条件语句和循环语句中。

上面程序中用 if yes and (t mod 7=0) then writeln；实现每行打印七个素数换行，程序中布尔变量 yes 为真，在逻辑上表示是素数；关系式（t mod 7=0) 的值为真时，表示该行输出素数已是 7 个；用 and 将这两个“条件”连起来是作一种布尔(逻辑)运算。

Pascal 共有四种逻辑运算符：

① and (与)　两条件都为 True 时，其结果值为 True；否则为 False；

② or (或)　 两条件中只要有一个为 True ；其结果值为 True；否则为 False；

③ xor (异或) 两条件的逻辑值不相同时，其结果值为 True；否则为 False；

④ not (非)　 条件为 True 时，其结果值为 False；否则为 True；(取反)

习题 3.1:

1.打印出 1 至 20 的平方数表。

2.打印出 100 至 200 之间的奇数。

3. 鸡兔同笼(用 for 循环程序完成)

4.一辆快车和一辆慢车开往同一地点，快车票价为 18 元，慢车票价为 13. 5 元，共售出400 张，共计 5940 元，求快车票和慢车票各多少张?.

5.求出能被 5 整除的所有四位数的和。

6.在下面式子中的二个□内填入一个合适的同样的数字，使等式成立。

□3*6528=3□*8256

7.有一个三位数，它的各位数字之和的 11 倍恰好等于它自身，请编程求出这个三位数。

8.在自然数中，如果一个三位数等于自身各位数字之立方和，则这个三位数就称为是水仙花数。如:$153=1^3+5^3+3^3$，所以 153 是一个水仙花数。求所有的水仙花数。

9.编程序打印出下列图案：

```
平行四边形          等腰三解形              菱形
    ******             *                    *
   ******             ***                  ***
  ******             *****                *****
 ******             *******                ***
******             *********                *
```

10.编程打印出如下图案：

```
    1
   222
  33333
 4444444
555555555
```

11.有三种明信片:第一种每套一张，售价 2 元；第二种每套一张，售价 4 元； 第三种每套 9 张，售价 2 元。现用 100 元钱要买 100 张明信片，要求每种明信片至少要买一套，问三种明信片应各买几套？请输出全部购买方案。

12.某人想把一元钱换成伍分、贰分、壹分这样的零钱， 在这三种零钱中每种零钱都至少各有一个的情况下，共有多少种兑换方案。并打出这些方案。

13.

14. 输出 100 以内的全部素数，要求每行显示 5 个。

15.A、B 两个自然数的和、差、积、商四个数加起来等于 243，求 A、B 两数。

16.百钱买百鸡：今有钱 100 元，要买 100 只鸡，公鸡 3 元一只，母鸡 1 元一只，小鸡 1 元 3 只，若公鸡、母鸡和小鸡都至少要买 1 只，请编程求出恰好用完 100 元钱的所有的买鸡方案。

第二节 repeat 循环

Repeat 循环是直到型循环。

试将上一节的例 3.1(打印出 1～20 的平方数表)程序改为 repeat 循环:

```
Program Exam31_1；
    Var a: byte；
    Begin
        a:=1；      writeln ( ' a ' : 8 ,    ' a*a ' : 8 ) ；
        repeat
          writeln ( a :8，a*a : 8)；
          inc(a)；                                    {改变 a 的值 }
        Until a＞20；
```

```
        Readln
    Emd.
```

程序中的 Repeat 循环格式为:

```
repeat
    循环体语句;
until 条件表达式;        {直到条件为真}
```

Repeat 循环首先执行由 Repeat 和 Until 括起来的循环体语句，然后检查 Until 后面的条件表达式：如果表达式结果为假，则继续执行循环体，接着继续检查 Until 后面的条件表达式，如此反复执行直到这个表达式结果为真时结束循环。Repeat 循环体语句必须有能改变 Until 后面条件表达式值的语句，并最终使这个条件表达式的值为真，使循环自动结束。

程序中 inc (a) 指令相当于 a :=a+1，常用的同类指令格式如下:

(1) inc(x) 等同 x:=x+1;
(2) inc(x, n) 等同 x:=x+n;
(3) dec(x) 等同 x:=x—1;
(4) dec(x，n) 等同 x:=x—n;

[例 3.10]求两个自然数 M 和 N 的最大公约数。

解：若自然数 a 既是 M 和约数，又是 N 的约数，则称 a 为 M 和 N 的公约数，其中最大的称为最大公约数。为了求得最大公约数，可以从最大可能的数（如 M 或 N）向下寻找，找到的第一个公约数即是最大公约数。

Pascal 程序：

```
Program ex310;
Begin
  a := N+1;
  Repeat
    a := a-1;
  Until (M mod a=0) and (N mod a=0);
  writeln(a);
  Readln;
End.
```

[例 3.11]校体操队到操场集合,排成每行 2 人,最后多出 1 人;排成每行 3 人,也多出 1 人;分别按每行排 4,5,6 人,都多出 1 人;当排成每行 7 人时,正好不多。求校体操队至少是多少人?

解:①设校体操队为 X 人,根据题意 X 应是 7 的倍数,因此 X 的初值为 7,以后用 inc(x,7)改变 X 值;

②为了控制循环, 用逻辑变量 yes 为真(True) 使循环结束;

③如果诸条件中有一个不满足, yes 的值就会为假(false)，就继续循环。

Pascal 程序:

```
program  Exam311;
var  x: word;  yes : boolean;
```

```
begin
    x:=0;
    repeat
       yes   :=true;   inc(x,7);
       if x mod 2 < > 1 then yes:=false;
       if x mod 3 < > 1 then yes:=false;
       if x mod 4 < > 1 then yes:=false;
       if x mod 5 < > 1 then yes:=false;
       if x mod 6 < > 1 then yes:=false;
    until yes;                                          {直到 yes 的值为真 }
    writeln('All =', x) ;   readln
end.
```

程序中对每个 X 值，都先给 Yes 赋真值，只有在循环体各句对 X 进行判断时，都得到“通过”（此处不赋假值）才能保持真值。

[例 3.12]从键盘输入一个整数 X（X 不超过 10000），若 X 的各位数字之和为 7 的倍数，则打印“Yes”，否则中打印“No”。

解：本题考察的是数字分离的方法，由于 X 的位数不定，所以以往的解法不能奏效，这是介绍一种取余求商法。

（1）用 X mod 10 分离出 X 的个位数字；

（2）用 X div 10 将刚分离的个数数字删除，并将结果送回给 X；

（3）重复（1）（2）直到 X＝0。

Pascal 程序：

```
Program ex12;
var x,a,s    : integer;
begin
   s := 0;
   repeat
      a := x mod 10;
      x := x div 10;
      s := s+a;
   until x=0;
   if s mod 7=0 then writeln(‘Yes’)
                else writeln(‘No’);
   Readln;
end;
```

[例 3.13]求 1992 个 1992 的乘积的末两位数是多少？

解：积的个位与十位数只与被乘数与乘数的个位与十位数字有关，所以本题相当于求 1992 个 92 相乘，而且本次的乘积主下一次相乘的被乘数，因此也只需取末两位参与运算就可以了。

Pascal 程序：

```
Program ex313;
var a,t : integer;
Begin
   a := 1;
   t := 0;
   repeat
      t := t+1;
      a := (a*92) mod 100;
   until t=1992;
   writeln(a);
   Readln;
End.
```

[例 3.14]尼科彻斯定理：将任何一个正整数的立方写成一组相邻奇数之和。

如: 3^3=7+9+11=27　　4^3=13+15+17+19=64

解:从举例中发现:

(1) n^3 正好等于 n 个奇数之和；

(2) n 个奇数中的最小奇数是从 1 开始的奇数序列中的第 m 个奇数，与 n 的关系为：m=n (n —1) / 2+1。

(3) 奇数序列中第 m 个奇数的值为 x，且 x= 2m—1，比如: n=3 时，m=3(3-1)/2+1=4，即 3 个奇数中最小的奇数是奇数序列中的第 4 个，它的值为 x=(2m-1)=7，所以：33=7+9+11。

(4) 从最小的奇数值 x 开始，逐个递增 2，连续 n 个,用 t 从 1 开始计数，直到 t=n 为止。

Pascal 程序:

```
Program Exam35；
Var n，m，x，t，s  :  integer；
Begin
   write(’input n:’)；    readln(n)；              {输入 N }
   m:=(n*(n-1) div 2)+1；                          {找到第 m 个奇数 }
   x:=2*m-1；  t:=1；        {算出第 m 个奇数的值 x，是所求的第一个}
   write(n’*’，n,’*’，n,’=’，x)；{输出第一个}
   s:=x;                                           {用 S 计算和 }
   if  n＞1  then
   Repeat
      inc(x,2)；                                   { 计算下一个奇数 }
      write (’+’，x) ；                            {加上下一个奇数 }
      inc (t )；  inc (s，x)；                     { 计个数并累加和 }
   Until t=n；                                     {直到 n 个 }
   Writeln (’=’，s ) ；
   Readln
End.
```

[例 3.15]猜价格：中央电视台的“幸运 52”栏目深受观众喜爱，其中的“猜商品价格”的节目更是脍炙人口，现在请你编一个程序模拟这一游戏：由计算机随机产生 200 至 5000 之间的一个整数，作为某件商品的价格，然后由你去猜是多少，若你猜的数大了，则计算机输出提示“Gao”，若你猜的数小了，则计算机输出提示“Di”，然后你根据提示继续猜，直到你猜对了，计算机会提示“Ok”，并统计你猜的总次数。

解：本题的游戏规则大家都清楚，要完成程序，必须把处理步骤理清：

（1）用随机函数 Random 产生 200 至 5000 之间的一个整数 X；

（2）你猜一个数 A；

（3）若 A＞X，则输出“Gao”；

（4）若 A＜X，则输出“Di”；

(5)若 A＝X 则输出“Ok”；

(6)重复(2)(3)(4)(5)直到 A=X。

Pascal 程序：

```
Program ex315;
Var t,X,a        : integer;
Begin
   Randomize;
   X := Random(4800)+200;
   t := 0;
   Repeat
      t := t+1;
      write(‘[‘,t,’] Qing cai yi ge zheng shu : ‘);
      readln(a);
      if a>x then writeln(‘Gao’);
      if a<x then writeln(‘Di’);
      if a=x then writeln(‘Ok’);
   Until A=X;
   Readln;
End.
```

习题 3.2

1.求两个自然数 M 和 N 的最小公倍数。（如果求三个或更多个数的最小公倍数呢？应如何解决）

2.小会议室里有几条相同的长凳,有若干人参加开会。如果每条凳子坐 6 人,结果有一条凳子只坐有 3 人;如果每条凳子坐 5 人,就有 4 人不得不站着。求会议室里有多少人开会,有多少条长凳?

3.某动物饲养中心用 1700 元专款购买小狗(每只 31 元)和小猫(每只 21 元)两种小动物。要求专款专用,正好用完, 应当如何购买?请输出所有方案。

4.某整数 X 加上 100 就成为一个完全平方数,如果让 X 加上 168 就成为另一个完全平方数。求 X?

5.某次同学聚会,老同学见面个个喜气洋洋,互相握手问好。参加此次聚会者每人都与老同学握了一次手,共握 903 次,试求参加聚会的人数？

6.用自然数300，262，205，167分别除以某整数A，所得到的余数均相同。求出整数A以及相除的余数？

7.1600年前我国的一部经典数学著作中有题：“今有物，不知其数，三三数之，剩二；五五数之，剩三；七七数之，剩二，问物几何。”求最小解。

8.编程求出所有不超过1000的数中，含有数字3的自然数，并统计总数。

9.阿姆斯特朗数：如果一个正整数等于其各个数字的立方和，则该数称为阿姆斯特朗数（也称自恋数），如$407=4^3+0^3+7^3$，试编程求出1000以内的所有阿姆斯特朗数。

第三节　While 循环

While循环是当型循环。

[例3.8] 前面第一章［例1.2］的鸡兔同笼,头30,脚90, 求鸡兔各几只？在此用下面方法编程求解。

解：设鸡为J只,兔为T只。已知头为H, 脚为F。

①让鸡的只数逐次加1进行递推计算，初始时J=0;

②计算兔的只数T=H-J;

③当总脚数(4*T+2*J)＜＞F就做 (J=J+1,T=H-J);

④当4*T+2*J=F时,说明所推算的J和T是正确的，应结束循环，并输出T, J。

Pascal程序:

```
Program  Exam38;
Const H=30;
      F=90;
Var  J,T : integer;
Begin
  J:=0;  T:=H－J;              {初始时让J从0开始计算 }
  While 4*T+2*J＜＞F  do      {当条件为真就做do后面的循环体 }
  begin
    inc(J);                 { 递推改变J值 }
    T:=H－J                  {计算兔的只数 }
  end;
  Writeln('T=',T,'  ':6,'J=', J ) ;
  Readln
End.
```

程序中采用While当型循环，While循环语句的格式为:

```
While 条件式 do 语句;
```

其中do后面的“语句”是被重复执行的,称为循环体;若循环体是多个语句, 必须用begin--end包起来成为复合语句。

While循环首先判断条件式，当条件式的值为真就执行do 后面的语句（循环体）。

While的循环体内也必须包含能改变控制变量取值语句, 影响条件式的值, 最终使条件式为false (假), 才能结束循环。

[例3.9] 输入任一的自然数A，B，求A ，B的最小公倍数。

解:这里采用适合计算机查找的方法：设D是它们的最小公倍数。先找出A，B当中的较大者并存放在A中，将较小者存放在B中，让D=A，当D能够整除B时，则D是所求的最小公倍数；当D不能整除B，就逐次地让D增加A。例如：A=18， B=12，步骤如下：

① 让D=A (D=18)

② 当(D mod B)＜＞0 为真时 (D不能整除B) 就做 D=D+A，重复②;

③ 当(D mod B)＜＞0 为假时结束循环，并输出D。

Pascal程序:

```
program  Exam39;
var a,b,d,t : word;
begin
   write('input a,b: '); readln(a , b);
   if a<b then
      begin
        t:=a; a:=b; b:=t
      end;
   d:=a;
   while  d mod b < >0  do                    {当条件为真时就做do后面的语句 }
     inc(d,a);
   writeln('[', a, ' , ' , b, ']=', d) ;
   readln
End.
```

Pascal语言的三种基本循环方式， for循环对循环范围有明确规定，且循环变量只能是递增加1或递减1自动计数控制；而repeat--until循环和while--do循环比较灵活，只要对条件表达式的值能控制满足一定要求就能组成循环，但在循环体中必须有改变循环变量值的语句，使条件判断（逻辑值）最终为True或flase，让循环能够终止。

[例3.10]求自然数A，B的最大公约数。

解:采用如下方法步骤:

(1)求A除以B的余数;

(2)当余数＜＞0就做n=a; a=b; b=n mod b，重复(1)和(2);

(3)当余数=0就结束循环，并输出b值。

比如a=18，b=12时，处理步骤为：

(1) = ，得余数为6;

(2) 此余数不为零 ，让a = 12，b = 6;

(3) 重复 = ， 得余数为0;

(4) 结束循环，输出6（余数为零时的b值即是所求的最大公约数）。

此方法称为辗转相除法求最大公约数。

Pascal程序:

```
program Exam310;
var a,b, n : word;
begin
```

```
    write('input a,b: ');     readln (a,b);
    write('(', a, ' , ' , b, ')=' ) ;
    while  a mod b < > 0  do
      begin
        n:=a;  a:=b;  b:=n mod b;
      end;
    writeln(b);
    readln
End.
```

[例3.11]将一根长为369cm的钢管截成长为69cm和39cm两种规格的短料。在这两种规格的短料至少各截一根的前提下，如何截才能余料最少。

解:设两种规格的短料分别为:

规格为69cm的x根，可在1至(369-39)/69范围循环取值;

规格为39cm的y根，用y = (369-69*X)/39)计算;

余料R=369-69*X-39*Y。

①设最小余料的初始值min=369;

②在X循环范围内,每一个X值都计算出对应的Y和R;

③如果R＜min，就将R存入min，x存入n，y存入m，记录余料最小时的x和y ;

④重复步骤②,当x值超出 （（369—39）/ 69）时结束循环。

Pascal程序:

```
program  exam311;
var x,y,r,min,n,m,a: integer;
begin
  min:=369;
  a:=(369-39) div 69; x:=1;
  while  x<=a  do
    begin
      y:=(369-69*x) div 39;
      r:=369-69*x-39*y;
      if r<min then
         begin
           min:=r; n:=x; m:=y
        end;
      inc(x);
    end;
  writeln('min=', min, '  x=', n, '   y=', m) ;
  readln
end.
```

在有些情况中，三种循环方法可以互相换用。

[例3.12]甲、乙、丙三人都是业余射击爱好者，在一次练习中他们枪枪中靶：甲射了八发子弹,取得225环成绩,乙射了七发,也取得225环;丙只射了六发,同样取得225环。下面是成绩表,请编程完成下表中空项的填数。

	射子弹数	中50环有几发	中35环有几发	中25环有几发	成绩（环）
甲	8				225
乙	7				225
丙	6				225

解:①设N为发射子弹数，只有8，7，6三个数字，正好又可用来代表甲、乙、丙；

②设A为中50环的子弹数，最小为0， 最大为(225 div 50=4)；

B为中35环的子弹数，最小为0，最大为(225 div 35=6)；

C为中25环的子弹数， C=N-A-B，但必须C＞0才可能正确；

③先让N=8， A取值(0～4)， B取值 (0～6)用循环逐个选定，在C＞0的情况下若能满足条件A*50+B*35+C*25=225就能确定一组填数。然后选N的下一数值,重复同样的过程。

Pascal程序:

```
program  exam312;
var a,b,c,n,s : integer;
begin
  writeln('n':3, 'a':3, 'b':3, 'c':3, 's':5) ;
  n:=8;
  while n<=6 do
    begin
      a:=0;
      while  a < = 4  do
        begin
          b:=0;
          while  b < = 6  do
            begin
              c:=n-a-b;
              if c>0 then
              begin
                 s:=50*a+35*b+25*c;
                 if s=225 then writeln(n:3,a:3,b:3,c:3,s:5);
               end;
             inc(b);
           end;
         inc(a);
       end;
       dec(n);
     end;
     readln
  end.
```

程序运行结果获得两组填数答案。如果改用for循环，程序将更加简明:

```
Program  Exam312_1;
Var a,b,c,n,s : Integer;
Begin
```

```
Writeln('N':3, 'A':3, 'B':3, 'C':3, 'S':5) ;
for n:=8 downto 6 do                    {N取值8，7，6，并分别代表甲、乙、丙 }
  for a:=0 to 4 do                      {中50环的可能范围 }
    for b:=0 to 6 do                    {中30环的可能范围 }
      begin
        c:=n-a-b;                       { 计算中25环的子弹数  }
          if c＞0 then begin                    {如果不是负数 }
             s:=50*a+35*6+25*c;          {计算总成绩 }
             if s=225 then writeln(n:3,a:3,b:3,c:3,s:5);
          end
        end;
  readln
End.
```

习题3.3

1.求S= 1-1/2 +1/3-1/4+1/5-1/6+ ……（求前N项的和）

2. Faibonacci数列前几项为：0,1,1,2,3,5,8,…,其规律是从第三项起，每项均等于前两项之和。求前30项，　并以每行5个数的格式输出。

3.小球从100高处自由落下,着地后又弹回高度的一半再落下。求第20次着地时，小球共通过多少路程?

4.某登山队员第一天登上山峰高度的一半又24米；第二天登上余下高度的一半又24米；每天均如此。到第七天,距山顶还剩91米。求此山峰的高度?

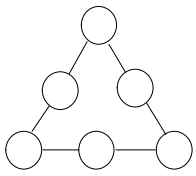
5.给出某整数N,将N写成因数相乘的形式。如：N=12,输出：12=1*2*2*3.

6.出售金鱼者决定将缸里的金鱼全部卖出。第一次卖出全部金鱼的一半加二分之一条;第二次卖出剩余的三分之一加三分之一条金鱼;第三次卖出余下金鱼的四分之一加四分之一条;第四次卖出余下的五分之一加五分之一条金鱼。还剩下11条金鱼。当然,出售金鱼时都是整数条,不能有任何破损。求缸里原有的金鱼数?

7.外出旅游的几位朋友决定次日早晨共分一筐苹果。天刚亮,第一个人醒来,他先拿了一个,再把筐里的八分之一拿走;第二个人醒来,先拿两个,再把筐里的八分之一拿走;第三个人醒来,先拿三个,再拿走筐里的八分之一;…每个人依次照此方法拿出各人的苹果,最后筐里的苹果全部拿完,他们每人所拿到的苹果数正巧一样多。求原先筐里的苹果数和人数。

8.图中由6个圆圈构成三角形,每条边上有三个圈，将自然数1--6 不重复地填入各圆圈位置上,使每条边圆圈上的数字之和相等,请编程输出所有的填法。

9．请编程显示出下面数字金字塔图形：



第四章　函数与过程

程序中往往需要把主要任务分成若干个子任务，每个子任务只负责一个专门的基本工作。每个子任务就是一个独立的子程序。Turbo Pascal 可以把函数和过程作为子程序调用。

第一节　函数

Pascal允许用户在程序中自己说明定义所需要的函数并在程序中调用这些函数。

[例4.1]编程找出由键盘任意输入五个整数中的最大整数。

解:设输入的五个整数为n1、n2、n3、n4、n5，为了便于处理，引入一个中间变量t1,按如下步骤处理：

①令t1=n1;

②将t1与n2比较，将两者中较大的数放入t1;

③将t1与n3比较，将两者中较大的数放入t1;

④将t1与n4比较，将两者中较大的数放入t1;

⑤将t1与n5比较，将两者中较大的数放入t1;

⑥经过以上5步处理后，t1即为5个数中最大者。

从上面规划的步骤看来，从步骤②到步骤⑤需处理的目标是相同的，因此我们可以设计一段子程序Max(x1,x2)，以找出x1和x2中最大的值并返回。

Pascal程序:

```
Program Exam41_a;
Var n1,n2,n3,n4,n5,t1 : integer;
Function max(x1,x2 : integer) : integer;
Begin
  If x1>x2 then Max := x1
           Else Max := x2;
End;

Begin
  Write(‘Input 5 numbers : ‘);
  Readln(n1,n2,n3,n4,n5);
  T1 := n1;
  T1 := Max(t1,n2);
  T1 := Max(t1,n3);
  T1 := Max(t1,n4);
  T1 := Max(t1,n5);
  Writeln(‘Max number : ‘,t1);
End.
```

从上例看出，引入函数实际上是将一个复杂的问题划分成若干个易于处理的子问题，将

编程化简的一种有效办法，而化简的方法是多种多样的，如前面已经做过求三个数中的最大数，所以可定义一个专门求三个数中最大数的函数(Max)。第一次用这个函数求出n1,n2,n3三个数中的最大数t1;第二次调用这个函数求出t1与n4,n5三个数中的最大数，也就是前三个数的最大数(已在t1中)和后面二个数再求一次，就得到五个数的最大数。因此，需要两次使用“求三个数中的最大数”，步骤如下：

①调用函数Max (n1, n2, n3)， 求出n1，n2，n3中的最大者 t1；

②调用函数Max (t1, n4, n5)，求出t1, n4, n5中的最大者t2；

③输出最大数 t2。

```
Program Exam41_b;
Var n1,n2,n3,n4,n5,t1: integer;

function Max(x1,x2,x3: integer): integer;                {自定义函数Max}
Var XX: integer;                                        {函数内部变量说明}
begin                                                      {函数体}
  if X1＞X2 then XX:＝X1
            else XX:＝X2;
  if X3＞XX then XX:＝X3;
  Max:＝XX
end;

Begin                                                      {主程序}
  Write('Input 5 numb:');
  Readln(n1,n2,n3,n4,n5);                                  {输入五个数}
  t1:＝Max(n1,n2,n3);                                {用函数求n1, n2, n3的最大数}
  t1:＝Max(n4,n5,t1);                                {用函数求n4, n5, t1 的最大数}
  Writeln('Max Number :', t1);
  Readln
End.
```

主程序中两次调用自定义函数。自定义函数的一般格式为：

```
function 函数名(形式参数表): 类型;    {函数首部}
局部变量说明部分;
begin
   语句系列;                                    {函数体 }
end;
```

函数中的形式参数接受调用函数时所传入的值，用来参与函数中的运算。

[例4.2]求任意输入的五个自然数的最大公约数。

解:⑴自定义一个专门求两自然数的最大公约数的函数GCD;

⑵调用自定义函数,第一次求前两个数的最大公约数;从第二次开始,用每次求得的最大公约数与下一个数再求两个数最大公约数,直到最后。本题共四次“求两个数的最大公约数”， 设输入的五个自然数分别是a1,a2,a3,a4,a5,采用如下步骤：

①求a1, a2两个数的最大公约数 → 存入a1；

②求a1, a3两个数的最大公约数 → 存入a1;
③求a1, a4两个数的最大公约数 → 存入a1;
④求a1, a5两个数的最大公约数 → 存入a1;
⑤ 输出 a1, 此时的a1已是五个数的最大公约数。
Pascal程序:

```
Program Exam42;
Var a1,a2,a3,a4,a5: integder;
function GCD(x,y: integer): integer;          {自定义函数 }
Var n:integer;
begin
  While  x mod y ＜＞0  do
  begin
    n:=x; x:=y; y:=n   mod   y
  end;
  GCD:=y
end;

Begin                                        {主程序 }
  Write('input 5 Numper:');
  readln(a1,a2,a3,a4,a5);                          {输入五个数}
  Write('(',a1,',',a2,',',a3,',',a4,',',a5,')=');
  a1:＝GCD(a1,a2);                                 {调用函数GCD }
  a1:＝GCD(a1,a3);
  a1:＝GCD(a1,a4);
  a1:＝GCD(a1,a5);
  Writeln(a1);
  readln
End.
```

函数的结果是一个具体的值, 在函数体中必须将所得到的运算结果赋给函数名; 主程序通过调用函数得到函数的运算结果。调用函数的一般格式为:

函数名 (实在参数表)

调用函数时, 函数名后面圆括号内的参数必须有确定的值, **称为实在参数。调用时即**把这些实际值传送给函数形参表中的相应形参变量。函数不是单独的语句, 只能作为运算赋值或出现在表达式中。

习题4.1

1. 数学上把从 1 开始的连续自然数相乘叫做阶乘。例如 把1*2*3*4*5 称作5的阶乘, 记为5! 。 编写一个求n!的函数, 调用此函数求: D＝

2.求从键盘输入的五个自然数的最小公倍数。

3.哥德巴赫猜想的命题之一是:大于6 的偶数等于两个素数之和。编程将6～100所有偶数表示成两个素数之和。

4.如果一个自然数是素数,且它的数字位置经过对换后仍为素数,则称为绝对素数,例如

13。试求出所有二位绝对素数。

第二节　自定义过程

自定义函数通常被设计成求一个函数值，一个函数只能得到一个运算结果。若要设计成能得到若干个运算结果,或完成一系列处理,就需要自定义“过程”来实现。

[例4.3] 把前面［例2.2 ］(输入三个不同的整数,按由小到大排序)改为下面用自定义过程编写的Pascal程序:

```
Program  exam43;
Var a,b,c: integer;
Procedure  Swap (var x,y: integer);            {自定义交换两个变量值的过程 }
Var t  : integer;
Begin                                          {过程体}
  t:=x; x:=y; y:=t                             {交换两个变量的值
end;

Begin                                          {主程序}
  Write('input a,b,c=');
  Readln(a,b,c);
  if a＞b then swap (a,b);                     {调用自定义过程}
  if a＞c then swap (a,c);
  if b＞c fhen swap (b,c);
  Writeln (a:6, b:6, c:6);
  Readln
End.
```

程序中Procedure Swap是定义过程名，从作用来看，过程与函数是相似的，都能将复杂的问题划分成一些目标明确的小问题来求解，只不过函数有值返回而过程则没有。自定义过程的一般格式如下:

```
Procedure 过程名 (形式参数表);        {过程首部}
  局部变量说明部分;
  begin
    语句部分;                          {过程体部分}
  end;
```

[例4.4]如果一个自然数除了1和本身,还有别的数能够整除它， 这样的自然数就是合数。例如15,除了1和15,还有3和5能够整除,所以15是合数。14,15,16是三个连续的合数,试求连续十个最小的合数。

解：从14,15,16三个连续合数中可看出,它们正好是两个相邻素数13和17 之间的连续自然数,所以求连续合数问题可以转化为求有一定跨度的相邻两个素数的问题。因此，求连续十个最小的合数可用如下方法：

①从最小的素数开始，先确定第一个素数A;

②再确定与A相邻的后面那个素数B;(作为第二个素数);

③检查A,B的跨度是度否在10 以上,如果跨度小于10,就把B 作为新的第一个素数A,重复作步骤②;

④如果A、B跨度大于或等于10，就打印A、B之间的连续10个自然数，即输出 A+1, A+2, A+3 ..., A+10。

Pascal程序:

```
Program  exam44;
var a,b,s,n: integer;
    yes: boolean;
procedure sub(x: integer;var yy: boolean);          {过程：求x是否为素数 }
var k,m: integer;                                   { 用yy逻辑值转出 }
begin
  k:=trunc(sqrt(x));
  for m:=3 to k do
  if odd(m) then
     if x mod m=0 then yy:=false;
end;

begin                                               {主程序 }
  b:=3;
  repeat
    a:=b;                                           {a 为第一个素数 }
    repeat
    yes:=true;
    inc(b,2);                                       {b是a后面待求的素数}
    sub(b,yes);                                     {调用SUB过程来确认b是否为素数 }
    if yes then s:=b-a;                             {如果b是素数，则求出跨度s  }
      until yes;
    until s > = 10;
    for n:=a+1 to a+10 do
       write(n:6);
    writeln;
    readln
  end.
```

程序中的过程SUB,用来确定b是否为素数。过程名后面圆括号内的变量是形式参数，简称为形参。过程SUB(x: integer; Var yy: boolean) 中的x是值形参,而前面冠有Var的yy是变量形参。值形参只能从外界向过程传入信息,但不能传出信息;变量形参既能传入又能传出信息。本程序过程SUB中的x是由调用过程的实在参数b传入值，进行处理后，不需传出;而yy是把过程处理结果用逻辑值传出，供调用程序使用。

试把［例4.3］程序中的过程 SWAP(Val x,y: integer)，将x,y前面的Var去掉，就变成了纯粹的值形参，就不能将过程所处理的结果传出去，也就无法得到处理后的结果，通过运

行程序比较，可以非常明显地看到值形参和变量形参的区别。

调用过程的格式为：　过程名（实在参数表）；

调用过程名后面圆括号内的实在参数与定义过程的形参表必须相对应，调用过程相当于一个独立语句，可单独使用。

［例4.5］将合数483的各位数字相加（4+8+3）=15，如果将483分解成质因数相乘：483=3*7*23，把这些质因数各位数字相加（3+7+2+3），其和也为15。即某合数的各位数字之和等于它所有质因数的各数字之和。求500以内具有上述特点的所有合数。

解：①设n为所要求的合数，让n在1～500间循环做以下步骤；

②用t1，t2分别累计合数n及n的质因数的各位数字之和，初值均为0；

③调用过程SUB3进行非素数判定，以布尔变量yes的真假值判定是否，yes的初值为true，如果为 (not true)非素数，就做步骤④，⑤，⑥；否则取新的n值，重复步骤③；

④调用SUB1，求n的各数字之和，传送给t1；

⑤调用SUB2，求n的各质因数，对于每个质因素都通过SUB1求它各位数字之和，将所有各质因数字传送给t2。

⑥如果t1=t2（各位数字之和等于它所有质因数的各数字之和），则输出此n。

PASCAL程序：

```
program exam45;
var n,t1,t2,tatol: integer;
    yes: boolean;
procedure sub1(x:integer; var t:integer);          {过程:分离x的各位数字 }
begin                                              {并求各位数字之和 }
  repeat
    t:=t+x mod 10;
    x:=x div 10;
  until x=0
end;

procedure sub2(x: integer; var t: integer);        {过程：分解质因数 }
var xx,tt:integer;
begin
  xx:=2;
  while x>1 do
    if x mod xx=0 then
      begin
        tt:=0;
        sub1(xx,tt);
        t:=t+tt;
        x:=x div xx
      end
    else inc(xx)
end;
```

```
procedure sub3(x: integer;var yy: boolean);      {过程：判断x是否为素数 }
var k,m: integer;
begin
  k:=trunc(sqrt(x));
  for m:=2 to k do
    if x mod m=0 then yy:=false;
end;

begin                                                {主程序}
   for n:=1 to 500 do
     begin
       t1:=0;t2:=0;
       yes:=true;
       sub3(n,yes);                    {调用过程求素数 }
       if not yes then                 {如果非素数就… }
         begin
           sub1(n,t1);                  {调用过程求n的各位数字之和 }
           sub2(n,t2);                  {调用过程求n的各质因数的数字之和 }
           if t1=t2 then write(n:6);                    {打印合格的合数 }
         end
     end;
   readln
end.
```

程序定义了三个过程SUB1，SUB2，SUB3，其中SUB2过程中又调用了SUB1。在过程中定义的变量或参数，只在本过程内部使用有效。这些变量称为局部变量。如SUB2中的xx只在SUB2中使用，属于局部变量。

习题：4.2

1.输入自然数n，求前n个合数（非素数），其素因子仅有2，3，或5。

2.自然数a的因子是指能整除a的所有自然数，但不含a本身。例如12的因子为：1,2,3,4,6。若自然数a的因子之和为b，而且b的因子之和又等于a，则称a,b为一对“亲和数” 。求最小的一对亲和数。

3.求前n个自然数的平方和，要求不用乘法。例如：3的平方不用3*3，可用3+3+3。

4.试用容积分别为17升、13升的两个空油桶为工具，从大油罐中倒出15升油来，编程显示出具体的倒油过程。

5.如果一个数从左边读和从右边读都是同一个数，就称为回文数。例如6886就是一个回文数，求出所有的既是回文数又是素数的三位数。

6. 任何大于2的自然数都可以写成不超过四个平方数之和。如：

$8=2^2+2^2$;　　$14=1^2+2^2+3^2$

由键盘输入自然数N（2 < N < 2000） ，输出其不超过四个平方数之和的表示式。

第五章　Pascal 的自定义数据类型

Pascal系统允许用户自定义的数据类型有：数组类型、子界类型、枚举类型、集合类型、记录类型、文件类型、指针类型。

第一节　数组与子界类型

［例5.1］总务室在商店购买了八种文具用品，其数量及单价如下表：

序号	1	2	3	4	5	6	7	8
品名	圆珠笔	铅笔	笔记本	订书机	计算器	三角板	圆规	文件夹
件数	24	110	60	16	26	32	32	42
单价	1.18	0.45	1.8	8.8	78.50	3.28	4.20	2.16

编程计算各物品计价及总计价。

解：表中有两组数据，设表示物品件数的一组为a，表示物品单价的一组为b。

a，b两组数据以序号为关联，具有相应的顺序关系。按如下方法处理：

①定义s，a，b三个数组，按相应顺序关系，给a，b赋值（件数和对应单价）；

②每读入一对数据（件数和对应单价），以同一序号的件数和对应单价计算出同一物品的计价：

```
  s[ i ]=a[ i ]* b[ i ] ;              { 用s[ i] 记入第i种物品的计价}
  t = t + s[ i ]                       {  用简单变量累加总计价 }
```

③循环做步骤②，做完后输出s数组所记入的各物品计价及总计价t。

Pascal程序：

```
Program  Exam51;
Var a: array[1..8] of integer;                {a数组为整数型}
    s,b: array[1..8] of real;                      {s和b数组为实数型}
    t: real;
    i: integer;
Begin
  t:=0;
  for i:=1 to 8 do                                        {输入并计算八种物品 }
    begin
      write('a[', i, ']=') ;
      Readln(a[ i ]) ;                              {输入单价}
      write('b[', i, ']=') ;
      readln(b[ i ]);                               {输入件数}
      s[ i ]:=a[ i ]* b[ i ];  t:=t+s[ i ]
    end;
  write('i':2, '  ':2);
```

```
  for i:=1 to 8 do                                    {打印物品序号}
     write(i:8);                                        {输出项宽度为8}
  writeln;
  write('a':2,'  ':2);                                  {输出项宽度为2}
  for i:=1 to 8  do                                  {打印物品件数a数组}
    write(a[ i ]:8);                                  {输出项宽度为8}
  writeln;                                              {换行}
  write('b':2,'  ':2);
  for i:=1 to 8  do                                  {打印物品件数b数组}
    write(b[ i ]:8:2);                               {输出项宽度为8,小数2位}
  writeln;                                              {换行}
  write('s':2,'  ':2);
  for i:=1 to 8  do                                  {打印物品计价s数组}
    write(s[ i ]:8:2);                               {输出项宽度为8,小数2位}
  writeln;                                              {换行}
  writeln('Totol=', t:8:2);                        {打印总价t}
  Readln
end.
```

输出语句为 write（实数：n：m） 的形式时，则输出该实数的总宽度为n，其中小数m位，此时的实数不以科学计数形式显示。

程序中用来表示如物品件数和物品单价等属性相同的有序数据，Pascal语言把它归为数组。数组成员（分量）称为数组元素。数组必须在说明部分进行定义：确定数组名，数组分量（元素）的个数及类型。一般格式有：

Var 数组名：array［下标类型］ of 数组元素类型 ；

本程序中a数组和b数组中8个元素的数据都是已知数据，可当作常量，用常量说明语句给数组元素赋初值，所以上面的程序Exam51可改为如下形式：

```
Program Exam51_1;
const  a: array[1..8] of integer
          =(24,110,60,16,26,32,32,42);                    {给a数组赋初值}
       b:array[1..8] of real
          =(1.18,0.45,1.80,8.8,78.50,3.28,4.20,2.16); {给b数组赋初值}
Var s: array[1..8] of real;
    t: real;
    i: integer;
Begin
  t:=0;
  for i:=1 to 8 do
    begin
      s[ i ]:=a[ i ]* b[ i ];  t:=t+s[ i ]
    end;
  write('i':2,'  ':2);
  for i:=1 to 8 do  write(i:8);
```

```
  writeln;
  write('a':2,'  ':2);
  for i:=1 to 8  do   write(a[ i ]:8:);
  writeln;
  write('b':2,'  ':2);
  for i:=1 to 8  do  write(b[ i ]:8:2);
  writeln;
  write('s':2,'  ':2);
  for i:=1 to 8  do  write(s[ i ]:8:2);
  writeln;
  writeln('Totol=', t:8:2);
  Readln
end.
```

数组常量说明格式为：

Const 数组名：array［下标类型］of 数组元素类型=（常量表）；

程序中对数组的输入、输出处理，常用循环语句控制下标，进行有序地直接操作每个数组元素。

［例5.2］编程输入十个正整数，然后自动按从大到小的顺序输出。

解：①用循环把十个数输入到A数组中；

②从A[1]到A[10]，相邻的两个数两两相比较，即：

A[1]与A[2]比，A[2]与A[3]比，……A[9]与A[10]比。

只需知道两个数中的前面那元素的标号，就能进行与后一个序号元素（相邻数）比较，可写成通用形式A[i]与A[i +1]比较，那么，比较的次数又可用1～(n - i)循环进行控制（即循环次数与两两相比较时前面那个元素序号有关）；

③在每次的比较中，若较大的数在后面，就把前后两个对换，把较大的数调到前面，否则不需调换位置。

下面例举5个数来说明两两相比较和交换位置的具体情形：

5	6	4	3	7	5和6比较，交换位置，排成下行的顺序；
6	5	4	3	7	5和4比较，不交换，维持同样的顺序；
6	5	4	3	7	4和3比较，不交换，顺序不变
6	5	4	3	7	3和7比较，交换位置，排成下行的顺序；
6	5	4	7	3	经过（1～（5-1））次比较后，将3调到了末尾。

经过第一轮的1～ (N-1)次比较，就能把十个数中的最小数调到最末尾位置，第二轮比较1～ (N-2)次进行同样处理，又把这一轮所比较的“最小数”调到所比较范围的“最末尾”位置；……；每进行一轮两两比较后，其下一轮的比较范围就减少一个。最后一轮仅有一次比较。在比较过程中，每次都有一个“最小数”往下“掉”，用这种方法排列顺序，常被称之为“冒泡法”排序。

Pascal程序：

```
Program  Exam52;
const  N=10;
Var a: array[1..N] of integer;                    {定义数组}
    i,j: integer;
```

```
procedure Swap(Var x,y: integer);                {交换两数位置的过程}
Var t:integer;
begin
  t:=x;  x:=y;  y:=t
end;
 Begin
   for i:=1 to N do                                        {输入十个数}
     begin
        write(i, ':');
        Readln(a[ i ])
     end;
   for j:=1 to N-1 do                                    {冒泡法排序}
     for i:=1 to N-j do                                  {两两相比较}
        if a[ i ] < a[i+1] then swap(a[ i ], a[i+1]);   {比较与交换}
   for i:=1 to N do                                       {输出排序后的十个数}
     write(a[ i ]:6);
   Readln
 end.
```

程序中定义a数组的下标类型为 1.. N ，这种类型规定了值域的上界和下界，是从一个有序类型范围取出一个区段，所以称为子界类型，子界类型也是有序类型。

例[5.3] 用筛法求出100以内的全部素数，并按每行五个数显示。

解：⑴ 把2到100的自然数放入a[2]到a[100]中（所放入的数与下标号相同）；

⑵ 在数组元素中，以下标为序，按顺序找到未曾找过的最小素数minp,和它的位置p（即下标号）；

⑶ 从p+1开始，把凡是能被minp整除的各元素值从a数组中划去（筛掉），也就是给该元素值置 0；

⑷ 让p=p+1，重复执行第②、③步骤，直到minp>Trunc(sqrt(N)) 为止；

⑸ 打印输出a数组中留下来、未被筛掉的各元素值，并按每行五个数显示。

用筛法求素数的过程示意如下（图中用下划线作删去标志）：

① 2 3 4 5 6 7 8 9 10 11 12 13 14 15…98 99 100　　{置数}

② 2 3 4 5 6 7 8 9 10 11 12 13 14 15...98 99 100　　{筛去被2整除的数 }

③ 2 3 4 5 6 7 8 9 10 11 12 13 14 15...98 99 100　　{筛去被3整除的数 }

……

2 3 4 5 6 7 8 9 10 11 12 13 14 15...98 99 100　　{筛去被整除的数 }

Pascal程序：

```
Program Exam53;
const  N=100;
type  xx=1 .. N;                                  {自定义子界类型xx（类型名）}
Var a: array[xx] of boolean;
```

```
    i,j: integer;
Begin
  Fillchar(a,sizeof(a),true);
  a[1] := False;
  for i:=2 to Trunc(sqrt(N)) do
     if a[I] then
         for j := 2 to N div I do
             a[I*j]:= False;
  t:=0;
  for i:=2 to N do
    if a[i] then
      Begin
        write(a[ i ]:5);  inc(t);
        if t mod 5=0 then writeln
      end;
  readln
End.
```

程序中自定义的子界类型，在用法上与标准类型（如integer）相同，只是值域上界、下界在说明中作了规定，而标准类型的值域由系统内部规定，其上界、下界定义是隐含的，可直接使用。例如：

Type integer= -32768 ... 32768; Pascal系统已作了标准类型处理，不必再作定义。

[例5.4]在一次宴会上，有来自八个不同国家的宾客被安排在同一张圆桌就坐。A是中国人，会讲英语；B是意大利人，他能讲西班牙语；C是英国人，会讲法语；D是日本人，能讲汉语；E是法国人，会讲德语；F是俄国人，懂意大利语；G是西班牙人，能讲日语；最后一个是德国人，懂俄语。编程序安排他们的座位，使他们在各自的座位上能方便地跟两旁的客人交谈。

解：①根据题目提供条件与数据，建立如下关系代码表：

国家名	中国	意大利	英国	日本	法国	俄国	西班牙	德国
宾客代码	A	B	C	D	E	F	G	H
语言代号	1	2	3	4	5	6	7	8
懂外语代号	3	7	5	1	8	2	4	6
总代码	A13	B27	C35	D41	E58	F62	G74	H86

表中总代码实际上是前三项代码的归纳：第一个字母表示哪国人；第二个数字表示本国语代号；第三个数字表示懂哪国外语。如A13，A表示中国人，1表示汉语（本国语），3表示会说英语。所以每个宾客的情况均用总代码（三个数据组成的字符串）表示；

②定义由8个元素组成的数组（NAME），元素类型为字符串类型（String）；

③元素的下标号影响各人座位关系，必须满足后一个元素的下标号应与前一个元素字符串中的第三个数据相同。例如：若第一个位置总代码为A13，则第二个位置应根据A13中最后的3，安排C35。即A与C可以交谈。以此类推。

用字符串处理函数COPY，截取字符串的第一个字母作为宾客代码打印，再取第三个字符，

用VAL将其转换成数字，将这个数字作为下标号，把这个下标号的元素安排在旁边（相邻）；

④重复步骤③的方法，安排其后的元素，直到八个数据全部处理完为止。

Pascal程序:

```
Program Exam54;
const  name :  array[1..8]of string              {定义字串类型数组并赋常量}
             =('A13','B27','C35','D41','E58','F62','G74','H86');
Var  i, code: integer;                           {整数类型}
     x:  1..8;                                   {子界类型}
     s  :  string;                               {字符串类型}
Begin
  s:=copy(name[1],1,1);                    {截取第一个元素字串的第一个字符}
  write(s:4);                              {确定第一个位置}
  s:=copy(name[1],3,1);                    {截取元素字串的第三个字符作为相邻}
  Val(s,x,code);                           {将字串s的值转换成数字存入 x}
  for i:=1 to 7 do                         {确定后面7个位置}
    Begin
      s:=copy(name[x],1,1);                      {找到相邻者的代码}
      write(s:4);                                {打印相邻者代码}
      s:=copy(name[x],3,1);                      {确定下一个相邻元素}
      Val(s,x,code);
    end;
  readln
End.
```

Pascal常用的字符串处理标准函数有7个：

设变量s，str，str1，str2均为字符串类型（string）｛多个字符｝；ch为字符类型（char）｛单个字符｝；

（1）copy（str，n，m）从字符串str的左边第n个开始截取m个字符；

如：copy（’Pascal’，3，2）的结果为'sc’；

（2）concat（str1，str2）将两个字串连接成为一个新的字串；

如：s:=str1+str2；同等于两串字符相加

（3）Length(str)求字串str的长度（字符个数）；

（4）chr(x) 求x（x为1…255整数)的ASII代码对应的字符；

如：chr（65）结果为 'A'。

（5）ord(ch)求字符ch对应的ASCII代码值；如 ord（'A'）结果为65；

（6）pos（str1，str2）求字串str1在字串中开始的位置；

如: pos（'sca'，'pascal'）结果为3；

（7）upcase(ch)将字符ch转为大写字母，如 upcase('a')结果为'A'；

Pascal常用的字符串处理标准过程有4个：

（1）Val(str，x，code)将数字型字串转为数字并存入变量x中；

如：Val(‘768',x,code),x值为768,code为检测错误代码，若code=0表示没有错误；

（2）str(n，s)将数字n转化为字串存入s中，如str(768，s)s的结果为 ’ 768’；

（3）insert(str1，str2，n)把字串str1插入在字串str2的第n个字符之前，结果在str2中；｛此过程中的str2为变量形参，具有传入传出的功能｝；

（4）delete(str，n，m)从字串str的第n个开始，删除m个字符，把剩余的字符存在str中，｛此过程中的str为变量形参，具有传入传出的功能｝；

[例5.5]一个两位以上的自然数，如果左右数字对称，就称为回文数，编程找出所有不超过6位数字的回文数，同时又是完全平方数的数。

如121是回文数，又是11的平方，所以是完全平方数。

解:①不超过6位数的完全平方数用循环在10～999范围产生（for i:=10 to 999）；

②将完全平方数（i*i)转成字串类型存入s中；

③逐个取s的左右字符，检查是否相同（对称），检查对数不超过总长度的一半；

④如果是回文数，就调用打印过程(Print)。

```
Program Exam55;
Var n, k, j ,t : integer;
    s  : string;                                    {字符串类型 }
    i:  longint;                                    {长整数类型 }
Procedure Print;                                      {打印过程(无形参)}
begin
  write(s  : 10);  inc(t);                          {打印s,  用t 计数 }
  if  t mod 6=0  then  writeln                  {打印6个换行 }
end;

Begin
  t:=0;
  for i:=10 to 999 do
    begin
      str(i*i,s);                                {将完全平方数转换成字串 }
      k:=length(s);                              {计算字串长度 }
      n:=k div 2;                                {计算字串长度的一半 }
      j:=1;
      while  j < = n  do                         {取左右字符检查是否对称 }
          if copy(s,j,1) < > copy(s,k+1-j,1) then j:=1000
          else inc( j ) ;                       {若不对称让j=1000,退出循环 }
      if j <1000 then Print                     { j <1000即是回文数,调打印 }
    end;
  writeln; writeln('Total=':8, t);                  {打印总个数 }
  readln
End.
```

习题5.1

1.裴波那契数列：数列1、1、2、3、5、8、13、21...称为裴波那契数列，它的特点是：

数列的第一项是1，第二项也是1，从第三项起，每项等于前两项之和。编程输入一个正整数N，求出数列的第N项是多少？（N不超过30）。

2.下面的竖式是乘法运算，式中P表示为一位的素数，编程输出此乘法竖式的所有可能方案。

```
  PPPP
x    P
------
 PPPPP
```

3.节目主持人准备从N名学生中挑选一名幸运观众，因为大家都想争当幸运观众，老师只好采取这样的办法：全体同学排成一列，由前面往后面依顺序报数1，2，1，2，…，报单数的同学退出队伍，余下的同学向前靠拢后再重新由前往后1，2，1，2，…报数，报单数者退出队伍，如此下去最后剩下一人为幸运观众。编程找出幸运观众在原队列中站在什么位置上？（N由键盘输入，N < 255）。

4. 1267*1267=1605289，表明等式右边是一个七位的完全平方数，而这七个数字互不相同。编程求出所有这样的七位数。

5. 校女子100米短跑决赛成绩如下表，请编程打印前八名运动员的名次、运动员号和成绩。（从第一名至第八名按名次排列）

运动员号	017	168	088	105	058	123	142	055	113	136	020	032	089	010
成绩(秒)	12.3	12.6	13.0	11.8	12.1	13.1	12.0	11.9	11.6	12.4	12.9	13.2	12.2	11.4

6. 求数字的乘积根。正整数的数字乘积这样规定：这个正整数中非零数字的乘积。例如整数999的数字乘积为9*9*9，得到729；729的数字乘积为7*2*9，得到126；126的数字乘积为1*2*6，得到12；12从数字乘积为1*2，得到2。如此反复取数字的乘积，直至得到一位数字为止。999的数字乘积根是2。编程输入一个长度不超过100位数字的正整数，打印出计算数字乘积根的每一步结果。输出格式如下：

（N=3486784401）

3486784401

516096

1620

12

2

7. 有一组13个齿轮互相啮合,各齿轮啮合的齿数依次分别为6,8,9,10,12,14,15,16,18,20,21,22,24, 问在转动过程中同时啮合的各齿到下次再同时啮合时,各齿轮分别转过了多少圈?

8. 集合M的元素的定义如下:

(1) 数1属于M;

(2) 若X属于M, 则A=2X+1, B=3X+1, C=5X+1, 也属于M;

(3) 再没有别的数属于M。(M={1,3,4,6,7,9,10,13,15,16...，如果M中的元素是按递增次序排列的,求出其中的第201,202和203个元素。

9. 一个素数,去掉最高位,剩下的数仍是素数;再去掉剩下的数的最高位,余留下来的数还是素数,这样的素数叫纯粹素数。求所有三位数的纯粹素数。

10. 自然数4,9,16,25等叫做完全平方数,因为2^2 =4, 3^2 =9, 4^2 =16,5^2 =25, 当某一对自然数相加和相减, 有时可各得出一个完全平方数。

例如：8与17这对自然数：　17+8=25　　17—8= 9

试编程,找出所有小于100的自然数对,当加和减该数对时，可各得出一个完全平方数。

第二节　二维数组与枚举类型

[例 5.6]假设四个商店一周内销售自行车的情况如下面表一所示，

自行车牌号	永久牌	飞达牌	五羊牌
第一商店	35	40	55
第二商店	20	50	64
第三商店	10	32	18
第四商店	38	36	28

表一

几种牌号自行车的单价如表二所示。求各店本周出售自行车的总营业额。

单价	元
永久牌	395
飞达牌	398
五羊牌	384

表二

解：①把表一看成是由行（每个店占一行）与列（每种牌号占一列）共同构成的数据组，按表格排列的位置顺序，用 A 数组表一各数据表示如下：

```
A[1,1]=35   A[1,2]=40   A[1,3]=55        {第一行共三个数据}
A[2,1]=20   A[2,2]=50   A[2,3]=64        {第二行共三个数据}
A[3,1]=10   A[3,2]=32   A[3,3]=18        {第三行共三个数据}
A[4,1]=38   A[4,2]=36   A[4,3]=28        {第四行共三个数据}
```

A 数组有 4 行 3 列，每个数组元素由两个下标号表示，这样的数组称为二维数组。

②表二的数据按排列顺序用 B 数组表示如下：

```
  B[1]=395      B[2]=398          B[3]=384
```

②B 数组有 3 个数据，用一维数组表示，下标号与表一中列的序号有对应关系。

③计算各店营业额并用 T 数组表示：

```
T[1]=A[1,1]*B[1]+A[1,2]*B[2]+A[1,3]*B[3]      {计算第一商店的营业额}
T[2]=A[2,1]*B[1]+A[2,2]*B[2]+A[2,3]*B[3]      {计算第二商店的营业额}
T[3]=A[3,1]*B[1]+A[3,2]*B[2]+A[3,3]*B[3]      {计算第三商店的营业额}
T[4]=A[4,1]*B[1]+A[4,2]*B[2]+A[4,3]*B[3]      {计算第四商店的营业额}
```

T 数组共有 4 个数据，为一维数组，下标号与商店号有对应关系。

④输出 T 数组各元素的值。

Pascal 程序：

```
Program  Exam56;
Var A: array[1..4,1..3] of integer;      {定义二维数组，整数类型}
    B: array[1..3] of integer;             {一维数组，3 个元素}
    T: array[1..4] of integer;             {一维数组，4 个元素}
   i,j: integer;
```

```
Begin
   for i:=1 to 4 do                        {输入表一的数据}
      Begin
         Write('A[',i,']: ');                  {提示输入哪一行}
         for j:=1 to 3 do  Read(a[i,j]);      {每行 3 个数据}
         Readln;                             {输完每行按回车键}
      end;
   for i:=1 to 3 do                          {输入表二的数据}
      Begin
         Write('B[', I ,']:');                       {提示第几行}
         Readln(B[ i ]);                     {输入一个数据按回车}
       end;
   for i:=1 to 4 do                           {计算并输出}
      Begin
         T[ i ]:=0;
         Write(' ':5,I:4);
         for j:=1 to 3 do
            Begin
              Write(A[i , j]:6);
              T[ i ]=T[ i ]+A[i , j]*B[j];
            end;
         Write(T[ i ]:8);
       end;
     Readln;
end.
```

程序中定义二维组方式与一维数组形式相同。二维数组的元素由两个下标确定。二维数组元素的格式如下：

数组名［下标 1，下标 2 ］

常用下标 1 代表数据在二维表格中的行序号，下标 2 代表所在表格中列的序号。

[例 5.7]输入学号从 1101 至 1104 的 4 名学生考试语文、数学、化学、英语、计算机六门课的成绩，编程求出每名学生的平均分，按每名学生数据占一行的格式输出。

Name	Chin	Math	Phys	Chem	Engl	Comp	Ave
1101	87	91	78	85	67	78	
1102	69	84	79	95	91	89	
1103	86	69	79	89	90	88	
1104	88	89	92	87	88	81	

解：根据题目所给数据及要求，定义如下数据类型：

①学生成绩：在数据表格中每人的成绩占一行，每行 6 列（每科占一列）；定义二维数组 s，各元素为实型；

②个人平均分：定义一维数组 av，各元素为实型；

③个人总分：是临时统计，为计算机平均分服务，用简单实型变量 t；
处理步骤为：
①用双重循环按行 i 按列 j 输入第 i 个人第 j 科成绩存入 s [i, j]；
②每读入一科成绩分，就累计到个人总分 t 中；
③输完第 i 个人的各科成绩，就计算出第 i 个人平均分并存入数组 av(i)中；
④重复上述步骤，直到全部学生的成绩处理完毕；
⑤用双重循环按行列形式输出完整的成绩表。
Pascal 程序：

```
Program Exam57;
const  NB=1101;  NE=1104;
Var   s: array[NB..NE,1..6] of real;   {定义二维数组(学生成绩)}
     av: array[NB..NE] of real;         {定义一维数组(平均成绩)}
      i: NB..NE;                            { i 为子界类型(学号范围)}
      j: 1..6;                              { j 为子界类型(课程范围)}
      t: eal;                               { t 为实数类型(计总成绩)}
Begin
  for  i:=NB  to  NE  do                    {用 i 控制处理一位学生成绩}
    begin
      t:=0;
      write(i, ':  ');
      for j:=1 to 6 do                      {输入并计算每人的 6 门成绩}
        begin
          read(s[i , j]);
          t:=t+s[i , j];                  {累加个人总分}
        end;
      av[ i ]:=t / 6;                       {求个人平均分}
      readln                                {输完 6 门分按一次回车键}
  end;
writeln;                                  {输出学生成绩表}
writeln('  ':5,' ************************************** ');
writeln('  ':5,'Name  Chin  Math  Phys  Chem  Engl  Comp  Ave');
writeln('  ':5,'----------------------------------------');
for i:=NB to NE do
  begin
    write(' ':5, i:4, '  ':2);       {输出学号}
    for j:=1 to 6 do
      write(s[i,j]:4:1, '  ':2);     {输出 6 门成绩}
    writeln(av[ i ]:4:1);             {输出平均分}
  end;
readln
End.
```

程序中的学生成绩用键盘输入方式赋给二维数组各元素，如果是少量已知数据，也可在

常量说明部分，直接给二维数组各元素赋常数，现将本例题改为如下程序：

```
Program Exam57_1;
Const NB=1101;  NE=1104;                              {定义常量}
Type  Cou=(Chin,Math,Phys,Chem,Engl,Comp);    {自定义枚举类型}
      Num=NB..NE;                                      {自定义子界类型}
Const s: array[Num,Cou] of real                  {定义二维数组并赋常数}
         =((87,91,78,85,67,78),
           (69,84,79,95,91,89),
           (86,69,79,89,90,88),
           (88,89,92,87,88,81));
Var  av: array[Num] of real;              {定义一维数组(平均成绩)}
     i: Num;                              { i 为子界类型(学号范围)}
     j: Cou;                              { j 为子界类型(课程范围)}
     t: real;                             { t 为实数类型(计总成绩)}
Begin
  for i:=NB to NE do                     {用 i 控制处理一位学生成绩}
    begin
      t:=0;
      for j:=Chin to Comp do             {计算每人的 6 门成绩}
        t:=t+s[i,j];                     {累加个人总分}
      av[ i ]:=t / 6;                    {求个人平均分}
    end;
  writeln;                                {输出学生成绩表}
  writeln(' ':5,'****************************************');
  writeln(' ':5,'Name  Chin  Math  Phys  Chem  Engl  Comp  Ave');
  writeln(' ':5,'----------------------------------------------');
  for i:=NB to NE do
    begin
      write(' ':5, i:4, ' ':2);          {输出学号}
      for j:=Chin to Comp do
        write(s[i,j]:4:1, ' ':2);        {输出 6 门成绩}
      writeln(av[ i ]:4:1);              {输出平均分}
    end;
 End.
```

程序说明部分定义了枚举类型。枚举类型常用自然语言中含义清楚、明了的单词（看成代码）来表示“顺序关系”，是一种顺序类型，是根据说明中的排列先后顺序，才具有 0，1，2…n 的序号关系，可用来作循环变量初值和终值，也可用来作数组下标。但枚举类型不是数值常量或字符常量，不能进行算术运算，只能作为“序号关系”来使用。

[例 5.8]从红(red)、黄(yellow)、兰(blue)、白(white)、黑(black)五种颜色的球中，任取三种不同颜色的球，求所有可能的取法？

解：①将五种颜色定义为枚举类型；

②a,b,c 都是枚举类型中取不同颜色之一；

③a 的取值范围从 red to black;

b 的取值范围从 red to black,但必须 a < > b;

c 的取值范围从 red to black,且必须 (a < > b) and (c < > b);

④每次打印取出的三个球的颜色，即第一个到第三个（for n:=1 to 3）

当 n=1：取 a 的值，根据 a 的“顺序”值输出对应颜色字符串；

当 n=2：取 b 的值，根据 b 的“顺序”值输出对应颜色字符串；

当 n=3：取 c 的值，根据 c 的“顺序”值输出对应颜色字符串；

⑤直至 a,b,c 的取值范围全部循环完毕。

Pascal 程序：

```
program ex58;
type color=(red,yellow,blue,white,black);
var a,b,c,dm: color;
    nn: 1..3;
    s: integer;
begin
  s:=0;
  for a:=red to black do
    for b:=red to black do
      if a < > b then
         for c:=red to black do
           if (c < > a) and (c < > b) then
              begin
                inc(s);  write(s:5);
                for nn:=1 to 3 do
                  begin
                    case  nn  of        {找每种球的颜色 “顺序”值}
                      1:  dm:=a;       {dm 是所取得的“顺序”值}
                      2:  dm:=b;
                      3:  dm:=c
                    end;
                    case  dm  of        {根据“顺序”值打印对应字串}
                      red   :   write('   red':9);
                      yellow :  write('yellow':9);
                      blue   :  write('  blue':9);
                      white  :  write(' white':9);
                      black  :  write(' blcak':9);
                    end;
                  end;
                writeln
              end;
  writeln;
```

```
  writeln('total num:':12, s:4);
  readln
end.
```

程序中的从 red 到 black 的顺序关系本来是不存在的，但经过枚举类型定义之后，就建立了这种“枚举”先后而产生的顺序排列关系。这种“关系”完全取决于类型说明时的位置排列。

[例 5.9] 新录 A、B、C 三个工人，每人分配一个工种，每个工种只需一人，经测试，三人做某种工作的效率如下表所示。如何分配三人的工作才能使他们工作效益最大？

工人＼工种	一	二	三
A	4	3	3
B	2	4	3
C	4	5	2

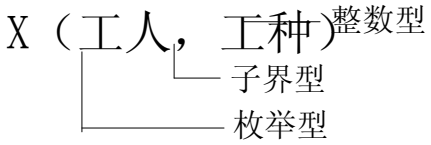
解：①定义各元素值为整数型的 X 数组，将表中的数据按行列关系作如下处理：

A 为第一行，将其三种工作效率(4,3,3)分别存入(x[A,1],x[A,2],x[A,3])；

B 为第二行，将其三种工作效率(2,4,3)分别存入(x[B,1],x[B,2],x[B,3])；

C 为第一行，将其三种工作效率(4,5,2)分别存入(x[C,1],x[C,2],x[C,3])。

在这里，x 数组第一个下标为枚举型，表示工人(A,B,C)；第二个下标为子界型，表示工种（一、二、三）：



②计算三人工作的总效率：S=x[A,i]+x[B,j]+x[C,k]

A 的工种 i：1 ～ 3 （用循环 for i:=1 to 3 ）；

B 的工种 j：1 ～ 3 （用循环 for j:=1 to 3 且 j< >i)；

C 的工种 k=6-i-j （工种代号总和为 6，减去两个代号就得到第三个)；

③将每次计算得到的 S 与“最大值”m 比较，(m 的初值为 0)，只要有大于 m 的 S 值即取代 m 原来的值，使之最大，同时用数组 dd 记录最大 S 值时的工种 i, j, k 值；

④当循环全部结束时，打印记录下来的每个人的工种。

Pascal 程序：

```
Program exam59;
type ma=(a,b,c);                          {定义枚举类型）
     wk=1..3;                            {定义子界类型}
Const x: array[ma,wk] of integer           {给 x 数组（二维）}
        =((4,3,3),(2,4,3),(4,5,2));       { 赋常量(表中值)}
      m: integer=0;                            {给 m 赋初值 0}
Var dd: array[wk] of wk;                  {用 DD 数组记忆工种号}
    i,j,k: wk;
    s: integer;
```

```
begin
  for i:=1 to 3 do
    for j:=1 to 3 do
      if j< >i then
        begin
          k:=6-i-j;
          s:=x[a,i]+x[b,j]+x[c,k];
          if s>m then
            begin
              m:=s;                    {记下最大效益}
              dd[1]:=i;                {记下最佳分配方案}
              dd[2]:=j;
              dd[3]:=k
            end
        end;
    for i:=1 to 3 do                       {输出}
      writeln(chr(64+i):8, dd[ i ]:8);
    writeln('最大效益 :' :12, m:4);
  readln
end.
```

输出语句中的 chr(64+i)是将(64+i)数值转换成对应的 ASCII 字符。

程序中用枚举类型表示工人代码（A,B,C），比较直观、清晰、易读好理解。

在程序中枚举类型都可以用数字序号来取代，下面程序用 1,2,3 代表工人 A,B,C：

```
program exam59_1;
type  n=1..3;
Const x: array[n,n] of integer
         =((4,3,3),(2,4,3),(4,5,2));
      m: integer=0;
Var dd: array[n] of n;
    i,j,k: n;
    s: integer;
begin
  for i:=1 to 3 do
    for j:=1 to 3 do
      if j < > i then
        begin
          k:=6-i-j;
          s:=x[1,i]+x[2,j]+x[3,k];
          if s > m then
            begin
              m:=s;
              dd[1]:=i;
```

```
                dd[2]:=j;
                dd[3]:=k
              end
          end;
      for i:=1 to 3 do
        writeln(chr(64+i):8,dd[ i ]:8, x[i,dd[ i ]]:8);
      writeln('最大效益 :' :12, m:4);
    readln
  end.
```

程序中的 x[i,dd[i]]是分配给 i 号工人做 dd[i]号工种的效率值，在这里，以数组元素值作为下标，称为下标嵌套。

[例 5.10] 下面是一个 3 阶的奇数幻方。

它由 1 到 3^2 的自然数组成一个 3*3 的方阵，方阵的每一行，每一列和两个对角线上的各数字之和都相等，且等于 n (n²+1) / 2（n 是方阵的行数或列数）。编程打印出 n 为 10 以内的奇数阶幻方。

6	1	8
7	5	3
2	9	4

解：仔细观察示例，有如下规律：

①在顶行中间填数字 1；{横坐标（行）X=1，纵坐标（列）Y=(n+1) / 2 }

②后继数放在前一个数的左上方；　{X=X-1; Y:= Y-1}

若超出了行，则认为是最后一行；{ if X<1 then X= n }

若超出了列，则认为是最后一列；{ if Y<1 then Y= n }

③若左上方已有数，则放在原数的正下方；{X=X+1, Y=Y }

④重复步骤②、③，直至填满方阵为止。

Pascal 程序：

```
Program Exam510;
 Uses  Crt;
 Var   a: array[1..10,1..10] of integer;
       x,y,xx,yy,s,n: integer;
 Begin
   Clrscr;                              {清屏}
   fillchar(a,sizeof ( a ), 0);         {将 a 数组各元素置 0}
   repeat
       write('Input Number Please!');  {提示输入 n}
       readln(n)
   until odd(n);
   s:=1;  x:=1;  y:=(n div 2)+1;  a[x,y]:=1;
   repeat
     inc(s);  xx:=x;  yy:=y;
     dec(x);  dec(y);
     if x<1 then  x:=n;
     if y<1 then  y:=n;
     if a[x,y]< >0 then  begin  x:=xx+1;  y:=yy  end;
```

```
    a[x,y]:=s;
  until s=n*n;
  for x:=1 to n do
    begin for y:=1 to n do write(a[x,y]:3);
          writeln
    end;
  repeat until keypressed;
End.
```

程序中 fillchar(a,sizeof （ a ）, 0) 是给 a 数组各元素置 0。Clrscr 是清屏。

习题 5.2:

1．输入四个学生考试五门功课，要求按个人总分从高到低排列输出二维成绩表格。(即每行有学号，五科成绩及总分)

2．杨晖三角形的第 n 行对应着二项式 n 次幂展开式的各个系数。例如第 3 行正好是 $(a+b)^3=a^3+3a^2b+3ab^2+b^3$ 展开式各项系数 1，3，3，1。

右图是 n 从 0～4 的杨晖三角形：

第一行 n=0,即 $(a+b)^0=1$，系数为 1；

第二行 n=1,即 $(a+b)^1=a+b$，系数为 1　1；

第三行 n=2,即 $(a+b)^2=a^2+2ab+b^2$，系数为 1　2　1；

编程输出 n 行的杨晖三角形。

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

3．下面是一个 4*4 的矩阵，它的特点是：(1)矩阵的元素都是正整数；(2)数值相等的元素相邻，这样，这个矩阵就形成了一级级“平台”，其上最大的“平台”面积为 8，高度（元素值）为 6。若有一个已知的 N*N 的矩阵也具有上面矩阵的特点，求矩阵最大“平台”的面积和高度。

```
6 6 6 7
1 6 3 7
1 6 6 7
6 6 7 7

N=4
Maxs=8 H=6
```

4．打印一个 n*n 的数字螺旋方阵。这个数字方阵的特点是：以左上角 1 开始向下，数字以外圈向里按自然数顺序转圈递增，一直到中心位置的 n^2 为止。例如 n =3:

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

第三节　集合类型

Pascal 系统把具有共同特征的同一有序类型的对象汇集在一起，形成一个集合，可将集合类型的所有元素作为一个整体进行集合运算。

[例 5.11]用随机函数产生 20 个互不相同的 40 到 100 的随机整数，然后按从小到大顺序打印。

解：按以下步骤处理：

①为使产生的随机整数互不相同。因此，每产生一个数，都要判断集合中已否包含，如

果没有包含，就放到集合中，并统计个数，直到 20 个。

②将集合中的数移到数组中，此题利用下标序号从小到大的特征进行映射排序打印。

Pascal 程序：

```
Program Exam511;
Uses  Crt ;
Var a: Array[40..100] Of boolean;
    dd: set Of 40..100;                {定义集合 dd}
    n: Integer;
Procedure  Init;                       {定义产生并处理随机数的过程}
Var i,m: Integer;
Begin
  n:=0;dd:=[ ];                        {集合 dd 初值为空}
  repeat
      begin
        Randomize;                     {将随机发生器作初始化处理}
        m:=Random(100);                {产生随机整数 m}
        if not (m in dd) and (m > 40 ) then
          begin
            dd:=dd+[m]; inc(n)         {把 m 放入集合 dd 中}
          end;
      end
  until n=20;
End;
Procedure Print;                       {定义打印过程}
Var i,j,k:Integer;
Begin
  fillchar(a,sizeof(a),false);         {将数组 a 的各元素置 false 值}
  For i:=40 To 100 Do
    if i in dd then a[ i ]:=true;      {以集合元素值为下标的数组元素赋真值}
  For i:=40 To 100 Do                  {以下标号为序(从小到大)输出}
    If a[ i ] Then  Write(i:4);        {输出 a 数组中元素值为真的下标号}
End;
Begin                                  {主程序}
  Clrscr;
  init;                                {产生随机数，并存入集合中}
  print;                               {打印}
  Repeat Until KeyPressed;
End.
```

程序中定义了集合类型 DD，集合的元素为子界类型。

定义集合类型的一般格式是：

Var　集合变量名:set of　元素的类型;

集合的值放在一对方括号中，各元素用逗号隔开，与排列的顺序无关，因此，[9,2,5]

和[2，5，9]的值相等，没有任何元素的集合是空集合，用[ ]表示。如果集合的元素是连续的，可用子界表示，如[5,6,7,8,9]可表示为[5 .. 9] 。

集合的赋值格式为：

集合变量名:= 集合表达式;

集合有以下几种运算：

1.集合的交、并、差运算：（设两个集合 a:=[1,2,4,6] 和 b:=[4,6,7,8] ）

①集合的并： a+b 即组合成新的集合(为[1,2,4,6,7,8])；

②集合的交： a*b 即将 a,b 集合中的公共元素组合成新的集合(为[4,6,])；

③集合的差： a-b 即在 a 中的元素去掉在 b 中出现的之后，所剩下的集合(为[1,2])。

2.集合的比较：

①相等：a=b，若两个集合中的元素个数相等，每个元素相同，则两个集合相等，比较结果为真（ture)，否则为假（false)；

②不等：a < > b 表示两个集合不相等；

③包含：a > = b 表示 a 集合包含 b 集合中的所有元素；

a < = b 表示 a 集合是 b 集合的子集。

3.集合的测试运算：检查某个数据在集合中，测试结果为 ture；不在集合中，测试结果为 false；例如：

6 in [8,6,9,4] 结果为 ture； { 6 在集合[8,6,9,4]中为真 }

2 in [8,6,9,4] 结果为 false； { 2 在集合[8,6,9,4]中为假 }

从程序 Exam511 的输出部分可看到，集合类型的值不能直接输出,要用测试方法进行输出或转换成数组元素的值。

[例 5.12]用集合进行筛法求 200 以内的素数。

解：①将[2..200]放入集合 S 中；

②取 S 中的第一个元素值 nxt,放入集合 P 中,同时将 S 中的凡是 nxt 的倍数的元素全部“划”去；

③重复步骤②，直至 S 集合为空；

④用测试运算打印 P 集合中全部元素值。

Pascal 程序：

```
Program Exam512;
Uses  crt;
const n=200;
var s,p: set of 2..n;                  { s,p 为集合类型}
   nxt,j,t: byte;
begin
   clrscr;
   s:=[2..n];                            {将[2..n]赋给 s}
   p:=[ ];
   nxt:=2;  t:=0;
   repeat
     while not(nxt in s) do
```

```
    nxt:=succ(nxt);                {后继函数}
  p:=p+[nxt];  j:=nxt;             {将 nxt 放入 P 中}
  while j<=n do
    begin
      s:=s-[j]; inc(j,nxt)         {筛掉 S 中的处理过的元素}
    end;
  if nxt in p then                 {用测试运算进行输出}
      begin
        inc(t);  write(nxt :6);
        if t mod 6=0 then writeln
      end;
  until s = [ ];
  readln
end.
```

集合内的元素个数不能超过 255 个，如果要用超过 255 个成员的集合类型求素数，必须用小集合的数组来表示大集合，即把大集合分成若干个小集合，每个小集合只是数组的元素，（数组元素为一个小集合）整个数组就是一个大集合。筛法运用在每个数组元素（小集合）中进行。

[例 5.13]将自然数 1--9 这九个数分成三组，将每组的三个数字拼成为三位数，每个数字不能重复，且每个三位数都是完全平分数。请找出这样的三个三位数。

解：①自定义函数 yes，用集合判定九个数字是否有重复，采用逆向思维，假设做邓了三个三位完全平方数：

将三个三位完全平方数分离成单个数字放入集合 dd 中，检查集合 dd，如果自然数 1～9 每个数恰好都在集合 dd 中，函数 yes 赋真（ture）；只要有一个不在集合中，九个数字没有占完集合中的九个位置，则必有重复，函数值为假（false），因为集合中对相同数字视为同一成员，如果有重复，则集合中不足 9 个成员（用测试运算）。

②程序用 11～31 平方产生三位的完全平方数。用循环方式每次取三个数为一组，存入 a 数组。

③对 a 数组的三位数调用自定义函数 yes 处理；

④如果函数 yes 值为真，就打印 a 数组中的三个数。

Pascal 程序：

```
Program exam513;
Uses Crt;
Var a: Array[1..3] Of Integer;
    i, j, k, x: Integer;
Function yes: Boolean;             {处理是否有重复数字}
Var i: Integer;
    d: Set Of  0 .. 9;             {集合元素为子界类型}
Begin
  d:=[ ];                          {集合的初值为空集合}
  For i:=1 To 3 Do      {将 a 数组中三个数分离成单个数并放入集合 d}
    d:=d+[a[ i ] Div 100, (a[ i ] Mod 100) Div 10, a[ i ] Mod 10];
```

```
  yes:=true;
  For i:=1 To 9 Do
    If Not ( i In d ) Then yes:=false;  {只要有一个不在集合中即为假}
End;
Begin
  writeln;
  for i:=11 to 29 do              {在三位完全平方数范围内循环推出三个数}
    Begin
      a[1]:=i*i;                  {第一个三位的完全平方数}
      for j:=i+1 to 30 do
        begin
          a[2]:=j*j;              {第一个三位的完全平方数}
          for k:=j+1 to 31 do
             begin
               a[3]:=k*k;         {第一个三位的完全平方数}
               If yes Then         {调用自定义 yes 函数结果为真就输出}
                    For x:=1 To 3 Do Writeln( x:8, ':', a[x]:8 );
             end
        end
     end;
  Repeat Until KeyPressed;
End.
```

习题 5.3

1.设计一个将十六进制数转换为十进制数的程序。

2.将自然数 1--9 数字不重复组成三个三位数，且三个数之比为 1∶2∶3。求出能满足条件的全部方案。

3.从键盘输入一个 20 位以内的自然数，然后将组成这个数的各位数字重新排列，得到一个数值为最小的新数，且新数的位数保持不变。打印出重新排列后的新数。

4.　现有五件物品，重量分别为 4、8、10、9、6.5 公斤,它们的价值分别为 12、21、24、17、10.5 元。有一个背包，装入物品总量不得超过 19 公斤，该选哪几件物品放入背包内使总价值最大?

第四节　记录类型和文件类型

前面介绍的数组类型和集合类型有一个共同点，那就是在同一个数组或集合中的各有元素都必须具有相同的类型。如果要处理如下表所示的学生档案卡片上的数据，各栏目的数据类型不一样（学号，姓名，成绩…）,需要用不同的类型表示。

学号	姓名	性别	出生年月	语文	数学	英语	平均分

为此，PASCAL 系统定义了记录类型，可用来表示不同类型的数据。

[例 5.14]建立一张学生情况表格，求出每位学生的平均成绩，并输出这张表格。

解：①定义记录类型：Date(表示日期记录，有三个域：day 日，mon 月，yea 年)；

Studa(表示学生情况记录，有六个域：nu 学号，na 姓名，dd 出生年月，se 性别，s 成绩，ave 平均分)；

②读入记录数据；

③计算学生的平均成绩；

④输出记录内容。

PASCAL 程序：

```
Program Exam514;
  Const n=2; m=3;                        {为了简单,人数 N=2 课目 M=3}
  Type  Date=Record                       {定义 Date(日期)记录类型}
              day: 1..31;                 {域名 day 表示天,为子界型(1..31)}
              mon: 1..12;                 {域名 mon 表示月,为子界型(1..12)}
              yea: 1970..1999;           {域名 yea 表示年,为子界类型}
            End;
        Studa=Record                      {定义 Studa(学生情况)记录类型}
               nu: string[5];             {域名 nu 表示学号,为字符串类型}
               na: string[8];             {域名 na 表示姓名,为字符串类型}
               dd: Date;                  {域名 dd 表示日期,为记录类型(Date)}
               se: char;                  {域名 se 表示性别,为字符类型}
               s: array[1..m] of real;    {域名 s 表示成绩,为数组类型}
               ave: real                  {域名 s 表示平均分,为实数类型}
             End;
      Sarr=array[1..n] of Studa;  {定义 Sarr 为数组类型,各元素为记录类型}
  Var  Stu: sarr;                        {变量 Stu 为数组(Sarr)类型}
Procedure rrd(var stu:sarr);            {定义输入和计算过程}
  var  i,k: integer;
       t: real;
       a: studa;                         {变量 a 为记录(Studa)类型}
  begin
    for k:=1 to n do
      begin
        with a,dd do                     {开域语句,打开当前记录 a 和 dd,进行以下操作}
          begin
            write(k:2,' nu: '); readln(nu);     {输入学号}
            write(k:2,' na: '); readln(na);     {输入姓名}
            write(k:2,' se: '); readln(se);     {输入性别}
            write(k:2,' day: '); readln(day);   {输入出生日}
```

```
            write(k:2,' mon: '); readln(mon);  {输入出生月}
            write(k:2,' yea: '); readln(yea);  {输入出生年}
            t:=0;
            for i:=1 to m do           {输入 m 科的成绩分}
                begin
                  write('s[',i,']='); read(s[ i ]);
                  t:=t+s[ i ]           {累加总分}
                end;
            readln;
            ave:=t/m;                 {计算平均分}
            stu[k]:=a                   {将当前记录存入 stu[k]中}
          end;
      end
  end;
Procedure print1;          {打印表格线}
  var  i: integer;
  begin
    for i:=1 to 60 do write('-');
    writeln;
                                                  type  记录类型名 = record
                                                                  域名：类型;
                                                                  …
                                                                  域名：类型;
                                                                end:
  end;
Procedure print2;         {打印表头和表格栏目}
  begin
    writeln(' ':18,'________ table _________' );
    print1;
    write(' num.', ' ':6, 'name', ' ':7, 'mm/dd/yy', ' ':4 );
    writeln('sex  ', 'Chin', ' ':2, 'math', ' ':2, 'Engl', ' ':2, 'vaer' );
    print1
  end;
Procedure print3(stu : sarr);   {打印记录数据}
  var i, j : integer;
  begin
    print2;
    for j:=1 to n do
      with stu[ j ], dd  do
        begin
          write(nu:5, na:9, '  ':8, mon:2, '/ ', day:2, '/ ',
                yea:4, '  ' );
          write(se:3, '  ' );
```

```
            for i:=1 to m do write(s[ i ]:6:1);
            writeln(ave:6:1);
            print1
          end
    end;
  Begin
    rrd(stu);
    print3(stu);
    readln
  end.
```

程序自定义 Date 记录类型，用来表示学生的出生年月日，含三个分量（称为域）：

day （日）　　为子界类型（1..31）;
mon（月）　　为子界类型（1..12）;
yea （年）　　为子界类型（1970..1999）;

自定义 Stuta 记录类型,用来表示学生情况，含六个域：

nu（学号）　　为字符类型（string [5] 为 5 个字符）;
na（学号）　　为字符类型（string [8] 为 8 个字符）;
dd（出生年月日） 是前面所定义的 date 记录类型 ;
se（性别）　　是 char 字符类型 ;
s（表示学生成绩）是数组类型，各元素为 real 实型 ;
ave（学生平均分）是 real 实型 ;

程序定义的数组 sarr 的每个元素为 Stuta 记录（每个记录相当于一张学生情况卡片，整个数组相当于全体学生的情况卡片）。

自定义记录类型的一般格式为：

访问记录中的分量，有两种方式：

① 记录名.域名　　例如对记录 a 中的 nu 赋值，可写成
　　a.nu:=1008; {将 1008 赋给记录 a 中的分量 nu}

② with 记录名 do 语句　　开域语句，with 后面可同时打开多个记录，
　　例如：with a, dd do 语句; { a, dd 都是记录名}

[例 5.15]利用记录类型将 N 个数由大到小排序，输出排序结果并显示每个数原来的位置序号。

PASCAL 程序：

```
Program Ex59;
  Type dd=Record                          {定义 DD 记录类型}
            ii:integer;                   {域名 ii 表示数}
            id:integer                    {域名 id 表示数 ii 的位置序号}
          End;
   Var a:array[1..100]of dd;
       i,j,k,n:integer;
       t:dd;
   Begin
     Write('Please Input Number of elements:');
```

```
read(n);
writeln('Input elements:');
for i:=1 to n do
  begin
    read(a[i].ii);
    a[i].id:=i;
    for j:=1 to i-1 do
      if a[j].ii<a[i].ii then
         begin
           t:=a[i];
           for k:=i-1 downto j do
             a[k+1]:=a[k];
           a[j]:=t;
         end;
   end;
 for i:=1 to n do
   begin
     write(a[i].ii:5,'(',a[i].id:2,')');
     if i mod 10=0 then writeln
   end;
   readln;writeln;
End.
```

PASCAL 系统自定义文件类型，可非常方便地实现对外存贮器的存取使用。常用的文件类型有顺序文件(File}和文本文件(Text)。

[例 5.16]建立一个由 1 到 50 连续整数为分量的顺序文件。

解：①定义顺序文件类型（file);
②说明文件变量名;
③用 assign 过程指派一个文件变量和实际磁盘文件名对应;
④建立一个实际文件;
⑤将变量的值写入文件中;
⑥关闭文件。

```
Program Exam516;
type fdata=file of integer;                    {定义文件类型}
var fd: fdata; i, k : integer;                 {fd 为文件变量}
begin
  assign(fd, 'a: files'); rewrite(fd);         {指派并建立文件}
  for I:=1 to 50 do write(fd, i ); close(fd); {写文件，最后关闭文件}
end.
```

[例 5.17]输出一个由 1 到 50 连续整数为分量的顺序文件。

解：①定义顺序文件类型（file);

②说明文件变量名；
③用 assign 过程指派文件变量和实际磁盘文件名对应；
④打开与文件变量对应的磁盘文件,并将文件分量指针设置在开头位置；
⑤读出指针所指的文件分量，赋给变量；
⑥关闭文件。

```
Program Exam517;
type fdata=file of integer;                    {定义文件类型}
var fd: fdata; i, k : integer;                 {fd 为文件变量}
begin
  assign(fd, 'files'); reset(fd);              {指派并恢复文件指针}
  for I:=1 to 50 do read(fd,i); close(fd);     {读文件，最后关闭文件}
end.
```

顺序文件在磁盘上以二进制形式存储，所以又称为二进制文件。

另一种常用的文件类型是文本文件，以 ASCII 码的形式存储，比较通用，可以用 DOS 命令 TYPE 显示文件内容。

[例 5.18]建立 由 50 个随机整数组成、文件名为 f1.dat 的 TEXT 文件。

解：①定义文件变量 f 为文本文件类型（TEXT）；
②指派 f 与磁盘（指定盘符 a:）上的 f1.dat 对应；
③建立文件；
④产生 1～50 个随机整数，写入文件中；
⑤关闭文件。

```
Program Exam518;
  const
    n=50;
var
    s: integer;                                {s 为 rr 型}
    f: text;                                   {f 为 text 类型}
    i: integer;
begin
  assign(f, 'a: f1.dat' );                     {用文件变量 f 与 a 驱磁盘上 f1.dat 文件对
应}
  rewrite(f);                                  {建立与 f 对应的实际文件名}
  randomize;                                   {初始化随机函数}
  for I:=1 to n do
    begin
      s:=random(99)+1;                         {产生一个 100 以内的随机整数赋给 s}
      write(f, s:6);                           {将 s 写入文件中}
      if I mod 5=0 then writeln(f)    {每行写 5 个数据}
    end;
  close( f )                                   {关闭文件}
end.
```

[例 5.19]读出 a 驱上磁盘文件 f1.dat，并输出打印文件内容。

解：①定义文件变量 f 为 text 类型；

②定义输出打印过程；

③定义读文件过程。

```
Program Exam519;
  const
    n=50;
var
    s: array[1..n] of integer;
    f: text;                              {文件变量 f 为文本类型(text)}
procedure pf ;                            {输出打印过程}
var i: integer;
  begin
    for I:=1 to n do
    begin
      write(s[ i ]:6);                    {打印 S 数组元素}
      if I mod 10=0 then writeln
    end;
  writeln
  end;
procedure rf ;                            {读出文件的过程}
  var i: integer;
  begin
    assign(f, 'a: f1.dat' ) ;             {指派 a 盘上 f1.dat 文件}
    reset(f ) ;                           {打开并恢复文件指针}
    i:=0;
    while not (eof ( f ) ) do             {当文件没有结束时就做以下语句}
      begin
        if not(eoln(f)) then              {如果不是行尾就读入数据}
             begin
               inc( i ); read(f, s[ i ]); {读入数据到数组 S 中}
             end
          else readln(f );                {否则就换行读}
      end;
    close(f )                             {关闭文件}
  end;
begin
  rf ;                                    {读文件}
  pf                                      {输出}
end.
```

文件类型常用指令表

操 作	格　　式	注　　释
指派文件	assign(文件变量, '路径: 文件名');	将文件变量指定为实际文件
建立文件	rewrite(文件变量);	建立一个文件
写文件	write(文件变量, 变量);	将变量值写入文件分量
打开文件	reset(文件变量);	打开文件，指针恢复到开头
读文件	read(文件变量,变量);	读文件分量值赋给变量
关闭文件	close(文件变量);	
文尾函数	Eof(文件变量)	判断所读文件结束时为 ture
行尾函数	Eoln(文件变量)	判断行结束时为 ture

习题 5.4

1.将 2 ～ 500 范围的素数存入 a: \ prin 中。

2.读出 a: \ prin 文件，并按每行 8 个数据的格式打印。

3.用随机函数产生 100 个三位数的随机整数，存入文件名为 ff.dat 中，然后读出该文件内容，进行从大到小排序，将排序后的数据按每行 10 个的格式显示，并存入文件名为 fff.dat 中。

4.将一段英文句子存入 Eng.dat 文件中（以每句为一行），然后读出并显示。（也可用汉语拼音代替英文句子）

5.建立 N 个学生档案卡，每张卡包含：编号、姓名、性别、年龄、三科成绩的等级（分 A、B、C 三档），编程示例输出显示。

第五节　指针类型与动态数据结构

前面所学的变量都具有共同的特点：系统依据程序说明部分获得变量名和类型信息，即为变量分配对应大小的存储空间，在程序执行过程中，各变量对应的存储空间始终存在且保持不变，这些变量均称为静态变量。

与静态变量对应的是动态变量，在程序执行过程中可以动态产生或撤消，所使用的存储空间也随之动态地分配或回收。为了使用动态变量。PASCAL 系统提供了指针类型，用指针变量（静态变量）来指示动态变量（存储地址变量）。下面介绍如何利用指针建立动态数据结构。

[例 5.19] 分别用简单变量和指针变量交换两个变量的值。

解：设两个变量 a,b 的值分别为 5, 8

(1)用简单变量交换：

```
Program Exam519;
  const  a=5; b=8;                                {常量}
  var c: integer;
  begin
```

```
  c:=a; a:=b; b:=c;                          {用简单变量交换}
```

Type　指针类型名 = ^ 基类型;

```
  writeln('a=':8, a, 'b=':8, b );
  readln
end.
```

(2)用指针变量交换：

Var　指针变量名 : ^ 基类型;

```
Program Exam519_1;
  type pon= ^ integer;                       {pon 为指针类型}
  var  a,b,c: pon;                           {a,b,c 为指针变量}
  begin
```

New（指针变量）;

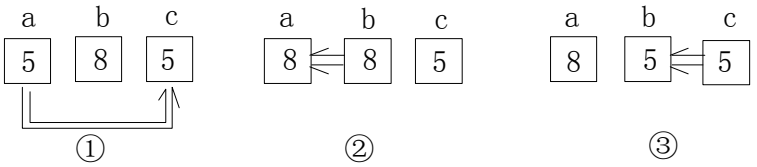
```
    new(a ); new(b ); new(c );               {开辟动态存储单元}
    a ^ :=5;  b ^ :=8;                       {给 a,b 指向的存储单元赋值}
    c:=a; a:=b; b:=c;                        {交换存储单元的指针}
    writeln('a=':8, a ^ , 'b=':8, b ^ );    {输出 a,b 所指单元的值}
```

Dispose（指针变量);

```
    readln
  End.
```

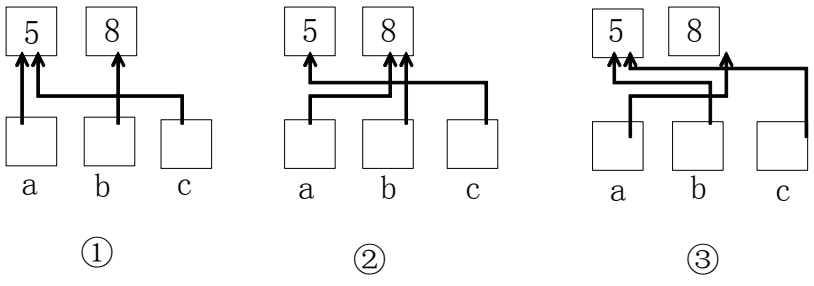
指针变量名 ^

第(1)种方法，直接采用变量赋值进行交换，（实际上给各存储单元重新赋值）其过程如下图所示：



（ 图表示 赋 值 ）

第(2)种方法，对各存储单元所保存的值并不改变，只是交换了指向这些单元的存储地址（指针值），可以简略地用如下图示说明：



(图中“—→”表示指向存储单元)

指针类型的指针变量 a,b 存有各指向单元的地址值，将指针交换赋值步骤为：

①将 c:=a (让 c 具有 a 的指针值)；

②将 a:=b (让 a 具有 b 的指针值)；

③将 b:=c (让 b 具有 c 的指针值)；

最后输出 a,b 所指向存储单元(a ^ 和 b ^)的值，此时的 a 指向了存储整数 8 的单元（即 a ^ = 8)，b 指向了存储整数 5 的单元（即 b ^ = 5)。存储单元的值没有重新赋值，只是存放指针值的变量交换了指针值。

程序 Exam519_1 Type pon= ^ integer; 是定义指针类型，一般格式为：

基类型就是指针所指向的数据元素的数据类型。也可以将类型说明合并在变量说明中直接定义说明：

例如 Exam519_1 中类型说明与变量可以合并说明为：

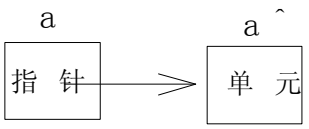
Var a,b,c: ^ integer; {与程序中分开说明的作用相同}

定义了指针变量的类型之后，必须调用 New 过程开辟存储单元，调用格式如下：

如果不再需要指针变量当前所指向的存储单元，可调用 Dispose 过程释放所占用的存储单元，Dispose 和 New 的作用正好相反，其格式为：

指针所指向的存储单元用如下形式表示：

程序中的 a ^ 表示指针变量 a 所指向的存储单元，与指针变量的关系如下图所示：



在程序中对所指存储单元（为 a ^）可以视作简单变量进行赋值计算和输出。

如：a ^ := 5

[例 5.20] 利用指针对数组元素值进行排序。

解：①定义一个各元素为指针类型的数组 a ：

②定义一个交换指针值的过程(swap)；

③定义一个打印过程(print)；

④定义过程(int)将数组 b 的值赋给 a 数组各元素所指向的各存储单元。

⑤过程 pixu 用交换指针值的方式，按 a 数组所指向的存储单元内容值从小到大地调整各元素指针值，实现“指针”排序；

⑥按顺序打印 a 数组各元素指向单元的值(a[i] ^)。

```
Program Exam520;
  const n=8;
        b: array[1..n] of integer
          =(44,46,98,86,36,48,79,71);
  type pon= ^ integer;
  var  a: array[1..n] of pon;
  Procedure swap(var p1, p2: pon);  {交换指针}
```

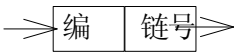
```
  var p: pon;
  begin
    p:=p1; p1:=p2; p2:=p
  end;
Procedure print;          {打印数组各元素指向单元(a[ i ] ^ )的值}
  var i: integer;
  begin
    for i:=1 to n do write(a[ i ] ^ :6);
    writeln; writeln;
  end;
Procedure int;          {将数组 b 的值赋给 a 数组各元素所指向的存储单元}
  var i: integer;
  begin
    for i:=1 to n do
     begin
       new(a[ i ]);
       a[ i ] ^ :=b[ i ];
     end;
    print;
  end;
Procedure pixu;                  {排序}
  var i,j,k: integer;
  begin
    for i:=1 to n-1 do
      begin
        k:=i;
        for j:=i+1 to n do
          if a[j] ^ < a[k] ^  then  k:=j;
        swap(a[k], a[ i ])
      end
  end;
Begin
 int;
 pixu;
 print;
 readln
End.
```

[例 5.21] 有 m 只猴子要选猴王，选举办法如下：

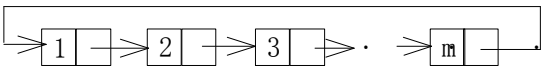
所有猴子按 1 ．．m 编号围坐成圆圈，从第一号开始按顺序 1，2，．．．，n 连续报数，凡报 n 号的退出到圈外。如此循环报数，直到圈上只剩下一只猴子即当选为王。用指针（环形链表）编程。

解：①让指针 pon 指向的单元为记录类型，记录内容含有两个域：

编号变量	链指针变量



②用过程 crea 建立环形链；



③用过程 king 进行报数处理：　　每报数一次 t 计数累加一次，当所报次数能被 n 整除，就删去该指针指向的记录，将前一个记录的链指针指向下一个记录。删去的指向单元（记录）应释放。直到链指针所指向的单元是同一单元，就说明只剩下一个记录。

④打印指向单元记录中编号域 (num)的值。

```
program Exam521;
 type  pon= ^ rec;                  {指向 rec 类型}
       rec=record                     {rec 为记录类型}
             num: byte;               {域名 num 为字节类型}
             nxt: pon                 {域名 nxt 为 pon 类型}
           end;
 var hd: pon;
    m, n: byte;
 procedure crea;                    {建立环形链}
    var s,p: pon;
         i: byte;
    begin
      new(s);  hd:=s;  s ^ . num :=1;  p:=s;
      for i:=2 to n do
        begin
          new(s);  s ^ . num :=i;  p ^ . nxt:=s;  p:=s
        end;
      p ^ . nxt :=hd
    end;
 procedure king;                    {报数处理}
   var p,q: pon;
       i, t: byte;
   begin
     p:=hd;  t:=0;  q:=p;
     repeat
       p:=q ^ . nxt;  inc(t);
       if t=n then
          begin
            q ^ . nxt :=p ^ . nxt;  dispose(p);  t:=0
          end
        else q:=p
     until p=p ^ . nxt;
    hd:=p
```

```
  end;
begin
  write('m, n='); readln(m, n);      {输入m只猴，报数到n号}
  crea;
  king;
  writeln('then king is :', hd ^ . num);
  readln
end.
```

习题 5.5

1.请将下列八个国家的国名按英文字典顺序排列输出。

China(中国)　　Japan(日本)　　Cancda(加拿大)　Korea(朝鲜)

England(英格兰)　France(法兰西)　American(美国)　India(印度)

2.某医院里一些刚生下来的婴儿 ，都还没有取名字，全都统一用婴儿服包装，很难区分是谁的小孩。所以必须建立卡片档案，包含内容有编号、性别、父母姓名、床号。实际婴儿数是变动的，有的到期了，家长要抱回家，要从卡片上注销；新的婴儿出生，要增加卡片，请编程用计算机处理动态管理婴儿的情况。

第六章 程序设计与基本算法

学习计算机语言不是学习的最终目的。语言是描述的工具，如何灵活地运用语言工具，设计和编写能解决实际问题的程序，算法是程序设计的基础。算法的作用是什么呢？著名数学家高斯（GAUSS）从小就勤于思索。1785 年，刚上小学二年级的小高斯，对老师出的计算题 S=1+2+3+…+99+100，第一个举手报告 S 的结果是 5050。班上的同学都采用依次逐个相加的“算法”，要相加 99 次;而小高斯则采用首尾归并，得出 S=（1+100）*50 的“算法”，只需加一次和乘一次，大大提高了效率。可见，算法在处理问题中的重要性。学习计算机编程，离不开基本算法。刚开始学习程序设计时，就应注重学习基本算法。

第一节 递推与递归算法

递推和递归是编程中常用的基本算法。在前面的解题中已经用到了这两种方法，下面对这两种算法基本应用进行详细研究讨论。

一、递推

递推算法是一种用若干步可重复的简单运算（规律）来描述复杂问题的方法。

[例 1] 植树节那天，有五位参加了植树活动，他们完成植树的棵数都不相同。问第一位同学植了多少棵时，他指着旁边的第二位同学说比他多植了两棵；追问第二位同学，他又说比第三位同学多植了两棵；…如此，都说比另一位同学多植两棵。最后问到第五位同学时，他说自己植了 10 棵。到底第一位同学植了多少棵树？

解：设第一位同学植树的棵数为 a1，欲求 a1，需从第五位同学植树的棵数 a5 入手，根据“多两棵”这个规律，按照一定顺序逐步进行推算：

①a5=10;

②a4=a5+2=12;

③a3=a4+2=14;

④a2=a3+2=16;

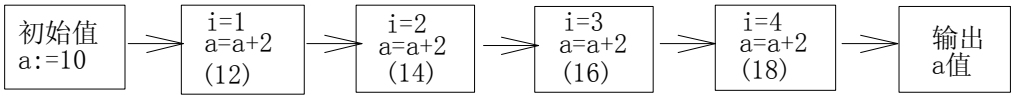
⑤a1=a2+2=18;

Pascal 程序：

```
Program Exam1;
  Var i, a: byte;
    begin
      a:=10;                {以第五位同学的棵数为递推的起始值}
      for i :=1 to 4 do     {还有 4 人，递推计算 4 次}
        a:= a+2;            {递推运算规律}
      writeln('The Num is', a);
```

```
    readln
  end.
```

本程序的递推运算可用如下图示描述：



递推算法以初始{起点}值为基础，用相同的运算规律，逐次重复运算，直至运算结束。这种从“起点”重复相同的方法直至到达一定“边界”，犹如单向运动，用循环可以实现。递推的本质是按规律逐次推出（计算）下一步的结果。

二、递归

递归算法是把处理问题的方法定义成与原问题处理方法相同的过程，在处理问题的过程中又调用自身定义的函数或过程。

仍用上例的计算植树棵数问题来说明递归算法：

解：把原问题求第一位同学在植树棵数 a1，转化为 a1=a2+2;即求 a2；而求 a2 又转化为 a2=a3+2；a3=a4+2；a4=a5+2;逐层转化为求 a2,a3,a4,a5 且都采用与求 a1 相同的方法；最后的 a5 为已知,则用 a5=10 返回到上一层并代入计算出 a4；又用 a4 的值代入上一层去求 a3;...,如此,直到求出 a1。

因此：

$$a_x=\begin{cases} 10 & (x=5) \\ a_{x+1}+2 & (x<5) \end{cases}$$

其中求 a_{x+1} 又采用求 a_x 的方法。所以:

①定义一个处理问题的过程 Num(x)：如果 X < 5 就递归调用过程 Num(x+1)；

②当递归调用到达一定条件(X=5)，就直接执行 a :=10，再执行后继语句，遇 End 返回到调用本过程的地方，将带回的计算结果(值)参与此处的后继语句进行运算（a:=a+2）；

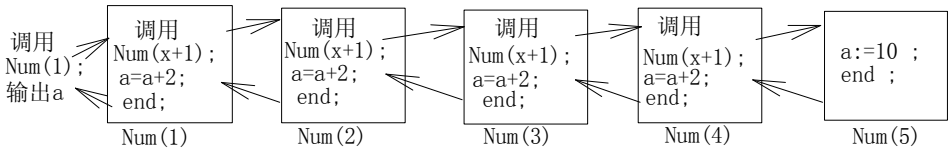
③最后返回到开头的原问题，此时所得到的运算结果就是原问题 Num(1)的答案。

Pascal 程序：

```
Program Exam1_1;
  Var a: byte;
  Procedure Num(x: integer);{过程 Num(x)求 x 的棵数}
    begin
      if x=5 then a:=10
         else begin
                Num(x+1); {递归调用过程 Num(x+1)}
                a:=a+2     {求(x+1)的棵数}
              end
    end;
  begin
    Num(1);      {主程序调用 Num(1)求第 1 个人的棵数}
    writeln('The Num is ', a);
    readln
```

end.

程序中的递归过程图解如下：



参照图示，递归方法说明如下：

①调用原问题的处理过程时，调用程序应给出具体的过程形参值（数据）；

②在处理子问题中，如果又调用原问题的处理过程，但形参值应是不断改变的量(表达式）；

③每递归调用一次自身过程，系统就打开一“层”与自身相同的程序系列;

④由于调用参数不断改变，将使条件满足（达到一定边界），此时就是最后一“层",不需再调用（打开新层），而是往下执行后继语句，给出边界值，遇到本过程的END，就返回到上“层”调用此过程的地方并继续往下执行;

⑤整个递归过程可视为由往返双向“运动”组成，先是逐层递进，逐层打开新的“篇章",(有可能无具体计算值)当最终递进达到边界，执行完本“层”的语句，才由最末一“层"逐次返回到上“层"，每次返回均带回新的计算值，直至回到第一次由主程序调用的地方，完成对原问题的处理。

[例 2] 用递归算法求 X^n 。

解：把 X^n 分解成: $X^0 = 1$ (n =0)

$X^1 = X * X^0$ (n =1)

$X^2 = X * X^1$ (n >1)

$X^3 = X * X^2$ (n >1)

…… (n >1)

$X^n = X * X^{n-1}$ (n >1)

因此将 X^n 转化为：

$$X^n=\begin{cases}1 & (n=0)\\ X*X^{n-1} & (n>1)\end{cases}$$

其中求 X^{n-1} 又用求 X^n 的方法进行求解。

①定义过程 xn(x,n: integer)求 X^n ;如果 n >1 则递归调用 xn (x, n-1) 求 X^{n-1} ;

②当递归调用到达 n=0，就执行 t t :=1, 然后执行本“层”的后继语句;

③遇到过程的 END 就结束本次的调用，返回到上一“层”调用语句的地方，并执行其后续语句 tt:=tt*x;

④继续执行步骤③，从调用中逐“层”返回，最后返回到主程序，输出 tt 的值。

Pascal 程序：

```
Program Exam2;
 Var tt, a, b: integer;
 Procedure xn(x, n: integer); {过程 xn(x, n)求 x^n }
   begin  if n=0 then tt:=1
            else begin
                   xn(x, n-1);   {递归调用过 xn(x,n-1)求 x^(n-1)}
```

```
                tt:=tt*x
              end;
    end;
  begin
    write('input x, n:');  readln(a,b);   {输入 a, b}
    xn(a,b);          {主程序调用过程 xn(a, b)求 a^b}
    writeln(a, '^', b, '=', tt);
    readln
  end.
```

递归算法，常常是把解决原问题按顺序逐次调用同一“子程序”(过程)去处理，最后一次调用得到已知数据，执行完该次调用过程的处理，将结果带回，按“先进后出”原则，依次计算返回。

如果处理问题的结果只需返回一个确定的计算值，可定义成递归函数。

[例 3]用递归函数求 x!

$$x!=\begin{cases}1 & (x=0)\\ x\times(x-1)! & \end{cases}$$

解：根据数学中的定义把求 x! 定义为求 x*(x-1)！,其中求(x-1)！仍采用求 x! 的方法，需要定义一个求 a! 的过程或函数，逐级调用此过程或函数，即：

(x-1)!= (x-1)*(x-2)！；

(x-2)!= (x-2)*(x-3)！；

……

直到 x=0 时给出 0!=1，才开始逐级返回并计算各值。

①定义递归函数：fac(a: integer): integer;

如果 a=0，则 fac:=1;

如果 a>0，则调用函数 fac:=fac(a-1)*a;

②返回主程序，打印 fac(x)的结果。

Pascal 程序：

```
Program Exam3;
 Var x: integer;
 function fac(a: integer): integer;  {函数 fac(a) 求 a !}
  begin
   if a=0 then fac:=1
     else fac:=fac(a-1)*a  {函数 fac(a-1)递归求(a-1) !}
  end;
  begin
    write('input x');  readln(x);
    writeln(x, '!=', fac(x));  {主程序调用 fac(x) 求 x !}
    readln
  end.
```

递归算法表现在处理问题的强大能力。然而，如同循环一样，递归也会带来无终止调用的可能性，因此，在设计递归过程（函数）时，必须考虑递归调用的终止问题，就是递归

调用要受限于某一条件，而且要保证这个条件在一定情况下肯定能得到满足。

[例 4]用递归算求自然数 A，B 的最大公约数。

解：求最大公约数的方法有许多种，若用欧几里德发明的辗转相除方法如下：

①定义求 X 除以 Y 的余数的过程；

②如果余数不为 0，则让 X=Y，Y=余数，重复步骤①，即调用过程；

③如果余数为 0，则终止调用过程；

④输出此时的 Y 值。

Pascal 程序：

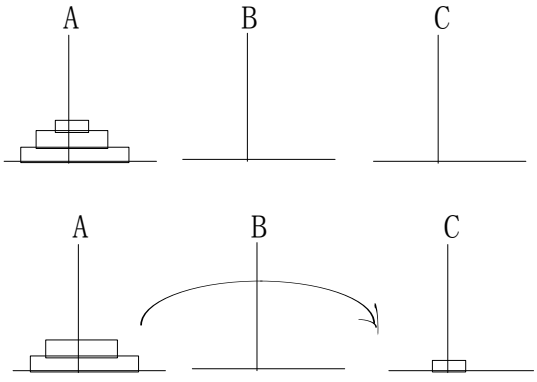
```
Program Exam4;
  Var a,b,d: integer;
  Procedure Gdd(x, y: nteger);{过程}
    begin
      if x mod y =0 then d :=y
       else Gdd(y, x mod y) {递归调用过程}
    end;
  begin
    write('input a, b='); readln(a, b);
    Gdd(a, b);
    writeln('(', a, ',', b, ')=', d );
    readln
  end.
```

简单地说，递归算法的本质就是自己调用自己，用调用自己的方法去处理问题，可使解决问题变得简洁明了。按正常情况有几次调用，就有几次返回。但有些程序可以只进行递归处理，不一定要返回时才进行所需要的处理。

[例 5] 移梵塔。有三根柱 A，B，C 在柱 A 上有 N 块盘片，所有盘片都是大的在下面，小片能放在大片上面。现要将 A 上的 N 块片移到 C 柱上，每次只能移动一片，而且在同一根柱子上必须保持上面的盘片比下面的盘片小，请输出移动方法。

解：先考虑简单情形。

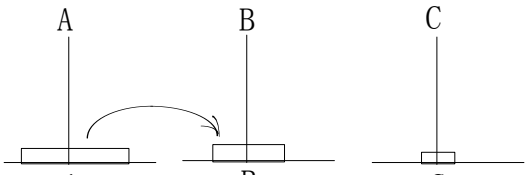
如果 N=3，则具体移动步骤为：



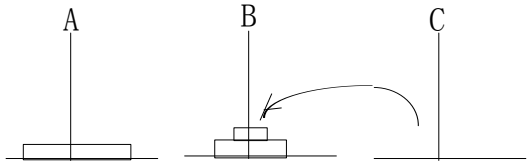
N=3 的初始状

移动方法和步

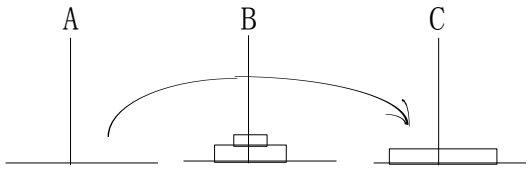
1. from A to C



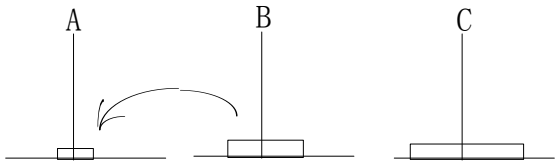
2.from A to B



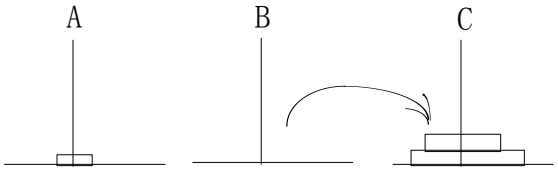
3.from C to B



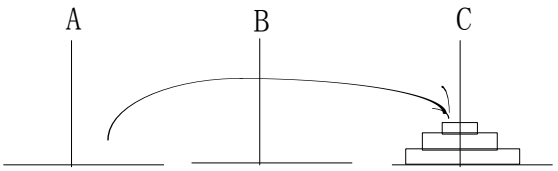
4.from A to C



5.from B to A

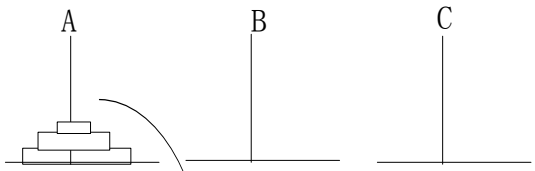


6.from B to C

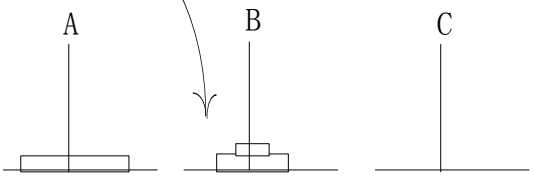


7. from A to C

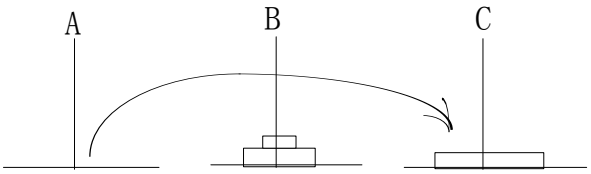
假设把第 3 步，第 4 步，第 6 步抽出来就相当于 N=2 的情况（把上面 2 片捆在一起，视为一片）：



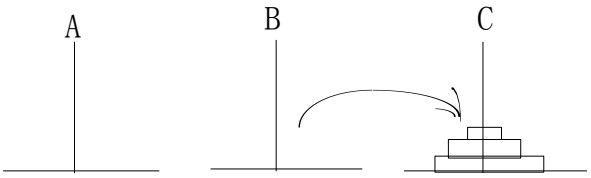
原 问 题 ： 欲 将 A 柱
C 柱 上 ，
记 为 （ A



1 ． 将 A 柱 上 的 （ N －
{ A 为 源 柱 ， B 为 |
渡 柱 ， 记 为 （ A



2．将 A 柱 上 剩 下 的
到 C 上 ； { 不 需



3．将 B 上 的 （N－1）
结 束 。 { B 为 源 柱 ，
柱 ， A 为 过 渡 柱 ，
（B，C，A）}

所以可按“N=2”的移动步骤设计：

①如果 N=0，则退出，即结束程序;否则继续往下执行;

②用 C 柱作为协助过渡，将 A 柱上的(N-1)片移到 B 柱上，调用过程 sub(n-1, a,b,c);

③将 A 柱上剩下的一片直接移到 C 柱上;

④用 A 柱作为协助过渡，将 B 柱上的（N-1）移到 C 柱上，调用过程 sub(n-1,b,c,a)。

Pascal 程序：

```
Program Exam65;
  Var x,y,z : char;
     N, k : integer;
  Procedure  sub(n: integer; a, c , b: char);
    begin
      if n=0 then exit;
      sub(n-1, a,b,c);
      inc(k);
      writeln(k, ': from', a, '-->', c);
      sub(n-1,b,c,a);
  end;
begin
  write('n='; readln(n);
  k:=0;
  x:='A'; y:='B'; Z:='C';
  sub(n,x,z,y);
  readln
end.
```

程序定义了把 n 片从 A 柱移到 C 柱的过程 sub(n,a,c,b)，这个过程把移动分为以下三步来进行：

①先调用过程 sub(n-1, a, b, c)，把(n-1)片从 A 柱移到 B 柱，C 柱作为过渡柱;

②直接执行 writeln(a, '-->', c)，把 A 柱上剩下的一片直接移到 C 柱上，;

③调用 sub(n-1,b,c,a)，把 B 柱上的(n-1)片从 B 移到 C 柱上，A 柱是过渡柱。

对于 B 柱上的(n-1)片如何移到，仍然调用上述的三步。只是把(n-1)当成了 n，每调用一次，要移到目标柱上的片数 N 就减少了一片，直至减少到 n=0 时就退出，不再调用。exit

是退出指令，执行该指令能在循环或递归调用过程中一下子全部退出来。

习题 6.1

1.过沙漠。希望一辆吉普车以最少的耗油跨越 1000 km 的沙漠。已知该车总装油量 500 升，耗油率为 1 升/ km，必须利用吉普车自己沿途建立临时加油站，逐步前进。问一共要多少油才能以最少的耗油越过沙漠？

2.楼梯有 N 级台阶，上楼可以一步上一阶，也可以一步上二阶。编一递归程序，计算共有多少种不同走法？

提示：如 N 级楼梯有 S(N)种不同走法，则有：

S(N)=S(N-2)+S(N-1)

3.阿克曼(Ackmann)函数 A(x,y)中，x,y 定义域是非负整数，函数值定义为：

A(x,y)=y+1　　(x = 0)

A(x,0)=A(x-1,1)　　(x > 0, y = 0)

A(x,y)=A(x-1, A(x, y-1))　　(x, y > 0)

设计一个递归程序。

4.某人写了 N 封信和 N 个信封，结果所有的信都装错了信封。求所有的信都装错信封共有多少种不同情况。可用下面公式：

Dn=(n—1) (D n—1 +D n—2)

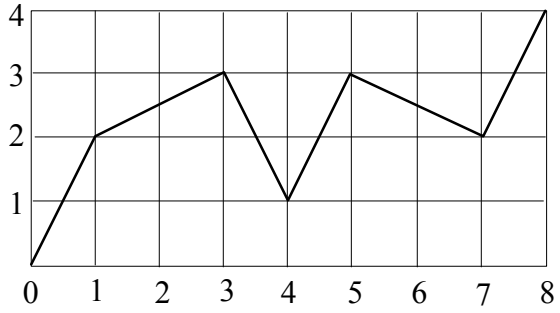
写出递归程序。

第二节　回溯算法

在一些问题求解进程中，有时发现所选用的试探性操作不是最佳选择，需退回一步，另选一种操作进行试探，这就是回溯算法。

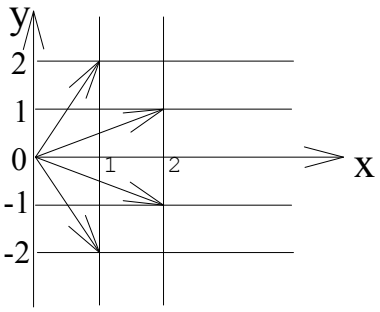
例[6.6] 中国象棋半张棋盘如下，马自左下角往右上角跳。现规定只许往右跳，不许往左跳。比如下图所示为一种跳行路线。编程输出所有的跳行路线，打印格式如下：

〈1〉 (0,0)—(1,2)—(3,3)—(4,1)—(5,3)—(7,2)—(8,4)



解：按象棋规则，马往右跳行的方向如下表和图所示：

x	1	2	2	1
y	2	1	-1	-2



水平方向用 x 表示；垂直方向用 y 表示。右上角点为 x=8, y=4, 记为(8, 4) ；用数组 tt 存放 x 方向能成行到达的点坐标;用数组 t 存放 y 方向能成行到达的点坐标;

①以(tt(K), t(k))为起点，按顺序用四个方向试探，找到下一个可行的点(x1, y1)；

②判断找到的点是否合理 (不出界)，若合理，就存入 tt 和 t 中;如果到达目的就打印，否则重复第⑴步骤;

③如果不合理，则换一个方向试探，如果四个方向都已试过，就退回一步(回溯)，用未试过的方向继续试探。重复步骤⑴;

④如果已退回到原点，则程序结束。

Pascal 程序：

```
Program Exam66;
   Const  xx: array[1..4] of 1..2 =(1,2,2,1);
          yy: array[1..4] of -2..2=(2,1,-1,-2);
   Var p: integer;
      t, tt : array[0..10] of integer;
   procedure Prn(k: integer);
       Var i: integer;
       Begin
           inc(p);  write('< ', p: 2, ' > ', '  ':4, '0,0');
           for i:=1 to k do
               write('— ( ', tt[ I ], ' , ', t[ I ], ')' );
           writeln
       End;
   Procedure Sub(k: integer);
       Var x1, y1, i: integer;
       Begin
           for I:=1 to 4 do
               Begin
                   x1:=tt[k-1]+xx[ i ];  y1:=t[k-1]+yy[ i ];
                   if not( (x1 > 8) or (y1 < 0) or (y1 > 4) ) then
                       Begin
                           tt[k]:=x1;  t[k]=y1;
                           if (y1=4) and (x1=8) then prn(k);
                           sub(k+1);
                       end;
```

```
                end;
        end;
    Begin
        p:=0;  tt[0]:=0;  t[0]:=0;
        sub(1);
        writeln( ‘ From 0,0 to 8,4  All of the ways are ’, p);
        readln
    end.
```

例[6.7] 输出自然数 1 到 n 所有不重复的排列，即 n 的全排列。

解：①在 1～n 间选择一个数，只要这个数不重复，就选中放入 a 数组中；

②如果这个数已被选中，就在 d 数组中作一个被选中的标记（将数组元素置 1 ）；

③如果所选中的数已被占用（作了标记），就另选一个数进行试探；

④如果未作标记的数都已试探完毕，那就取消最后那个数的标记，退回一步，并取消这一步的选数标记，另换下一个数试探，转步骤①；

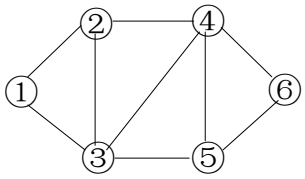
⑤如果已退回到 0，说明已试探全部数据，结束。

Pascal 程序：

```
Program Exam67;
    Var p,n: integer;
        a,d: array[1..500] of integer;
    Procedure prn (t : integer);
        Var i: integer;
        Begin
            write(‘ < ‘, p:3, ’ > ‘, ’  ‘:10);
            for I:=1 to t do
                write(a[ I ]:4);
            writeln;
        end;
    Procedure pp(k: integer);
        var x: integer;
        begin
            for x:=1 to n do
                begin
                    a[k]:=x; d[x]:=1;
                    if k < n then pp(k+1)
                    else
                        begin
                            p:=p+1;
                            prn(k);
                        end;
                end;
        end;
```

```
Begin
    write('Input n='); readln(n);
    for p:=1 to n do d[p]=0;
    p:=0;
    pp(1);
    writeln('All of the ways are ', p:6);
End.
```

例[6.8] 设有一个连接 n 个地点①—⑥的道路网，找出从起点①出发到过终点⑥的一切路径，要求在每条路径上任一地点最多只能通过一次。



解：从①出发，下一点可到达②或③，可以　　分支。具体步骤为：

⑴假定从起点出发数起第 k 个点 Path[k]，　　如果该点是终点 n 就打印一条路径；

⑵如果不是终点 n，且前方点是未曾走过的　　点，则走到前方点，定(k+1)点为到达路径，转步骤⑴；

(3)如果前方点已走过，就选另一分支点；

(4)如果前方点已选完，就回溯一步，选另一分支点为出发点；

(5)如果已回溯到起点，则结束。

为了表示各点的连通关系，建立如下的关系矩阵：

第一行表示与①相通点有②③，0 是　　结束标志;以后各行依此类推。

$$Roadnet=\begin{bmatrix} 2 & 3 & 0 & & \\ 1 & 3 & 4 & 0 & \\ 1 & 2 & 4 & 5 & 0 \\ 2 & 3 & 5 & 6 & 0 \\ 3 & 4 & 6 & 0 & \\ 4 & 5 & 0 & & \end{bmatrix}$$

集合 b 是为了检查不重复点。

```
Program Exam68;
const n=6;
    roadnet: array[1..n, 1..n] of 0..n=( (2,3,0,0,0,0),
                                         (1,3,4,0,0,0),
                                         (1,2,4,5,0,0),
                                         (2,3,5,6,0,0),
                                         (3,4,6,0,0,0),
                                         (4,5,0,0,0,0) );
var b: set of 1..n;
   path: array[1..n] of 1..n;
   p: byte;
 procedure prn(k: byte);
 var i: byte;
 begin
   inc(p); write('<', p:2, '>', ' ':4);
```

```
    write (path[1]:2);
    for I:=2 to k do
     write ('--', path[ i ]:2);
    writeln
  end;
  procedure try(k: byte);
  var j: byte;
  begin
```

1	2	3	4	5
6	X	8	9	10
11	12	13	14	15

```
    j:=1;
    repeat
    path[k]:=roadnet [path [k-1], j ];
    if not (path [k] in b) then
       begin b:=b+[path [k] ];
        if path [k]=n then prn (k)
         else try(k+1);
        b:=b-[path [k] ];
       end;
    inc(j);
    until roadnet [path [k-1], j ]=0
  end;
begin
  b:=[1]; p=0; path[1]:=1;
  try(2);
  readln
end.
```

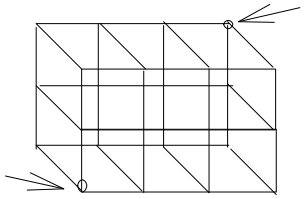
习题[6.2]

1. 有A,B,C,D,E五本书，要分给张、王、刘、赵、钱五位同学，每人只能选一本。事先让每个人将自己喜爱的书填写在下表中。希望你设计一个程序，打印分书的所有可能方案，当然是让每个人都能满意。

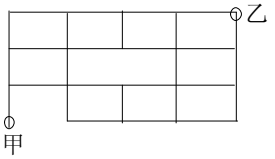
	A	B	C	D	E
张			Y	Y	
王	Y	Y			Y
刘		Y	Y		
赵				Y	

钱 Y Y

2. 右下图所示的是空心框架,它是由六个单位正方体组成,问:从框架左下外顶点走到右上内顶点共有多少条最短路线?



3.城市的街道示意图如右:问从甲地去到乙地可以有多少条最短路线?



4.有 M×N 张(M 行, N 列)邮票连在一起,

但其中第 X 张被一个调皮的小朋友控掉了。上图是 3×5 的邮票的形状和编号。从这些邮票中撕出四张连在一起的邮票,问共有多少种这样四张一组的邮票?注:因为给邮票编了序号,所以 1234 和 2345 应该看作是不同的两组。

5.有分数$\frac{1}{2}$，$\frac{1}{3}$，$\frac{1}{4}$，$\frac{1}{5}$，$\frac{1}{6}$，$\frac{1}{8}$，$\frac{1}{10}$，$\frac{1}{12}$，$\frac{1}{15}$， 求将其中若干个相加的和恰好为 1 的组成方案，并打印成等式。例如：

〈1〉 $\frac{1}{2}+\frac{1}{3}+\frac{1}{6}=1$

〈2〉 ...

6.八皇后问题。在 8*8 的国际象棋盘上摆上 8 个皇后。要求每行，每列，各对角线上的皇后都不能互相攻击，给出所可能的摆法。

第七章　数据结构及其应用

数字，字符，声音，图像，表格等信息，均可输入计算机中进行处理。在计算机科学中，象这种能输入到计算机中并被计算机程序处理的信息，都可称为数据。

数据的基本单位是数据元素。数据之间存在有线性与非线性两种基本的逻辑结构，同时在存储结构上还有顺序和链式之分。

数据结构则是研究数据元素的逻辑结构，存储结构和与之有关的各种基本操作的一门学科。作为一个程序设计者，应当掌握好数据结构的有关知识，在解题时针对问题的特点，选择适当的数据结构，并构造算法，编出优美高效的好程序。

本章将介绍一些线性的数据结构及其基本操作。

第一节　线性表

“线性表”是指由有限多个类型相同的数据元素组成的集合，它有以下的特点：

(1) 有唯一的头结点(即第一个数据元素)和尾结点(即最后一个数据元素)；

(2) 除结点外，集合中的每个数据元素均只有一个前驱；

(3) 除尾结点外，集合中的每一个数据元素均只有一个后继。

“线性表”是一种运用非常广范的数据结构。

例一、某旅馆有 100 个房间，以 1 到 100 编号，第一个服务员来了，他将所有的房门都打开，第二个服务员再把所有编号是 2 的倍数的房门都关上，第三个服务员对编号是 3 的倍数的房门原来开的关上，原来关上的打开，此后的第四，五...服务员均照此办理。问第 100 个服务员走进后，有哪几扇门是开着的。

解：Pascal 程序：

```
Program lt7_1_1；
uses crt；
var door：array[1..100] of boolean；  1 到 100 号房门状态，false--关，true--开
    i，j：integer；
begin
  clrscr；
  fillchar(door，sizeof(door)，true)；    第一个服务员打开全部房门
  for i：=2 to 100 do              i 表示服务员号码
    for j：=1 to 100 div i do
      door[i*j]：=not door[i*j]；  对房号为 i 的倍数的房门进行相反处理
  write('The code of opening door is ：  ')；
  for i：=1 to 100 do
    if door[i] then write(i，' ')；
end.
```

分析：(1)这里用 door[1..100]来存储 1 到 100 号房门的开关状态，即是一种线性表，其中：door[1]可以看成是头结点，而 door[100]是尾结点；同时由于数组在内存中是按顺序占有一片连续的内存空间，因此这种存储结构即是顺序存储结构。

(2)这里用布尔变量 true 和 false 分别表示开和关两种状态，对某一房门进行相反处理时只要取反(not)即可，比如：若 door[10]为 false，not door[10]则为 true。

例二、插入排序：在一个文本文件中存放的 N 个人的姓名，文本文件的格式为：第一行为 N，以下第二至第 N+1 行分别是 N 个人的姓名(姓名不重复，由英文字母组成，长度不超过 10)，请编一个程序，将这些姓名按字典顺序排列。

解：Pascal 程序：

```
Program lt7_1_2;
uses crt;
type point=^people;     定义结点类型
       people=record
         name：string[10];     name--数据域，存放姓名
         next：point;       next--指针域，存放后继结点的地址
       end;
var head：point;
     n：integer;

procedure init;     初始化
  begin
    new(head)；head^.next：=nil;  定义头结点，初始链表为空
  end;

procedure insert(p：point);  将 P 指向的结点插入以 head 开头的线性表中
  var q1，q2：point;
  begin
    if head^.next=nil then head^.next：=p  将 P 指向的结点插入空链表
        else
          begin
            q1：=head；q2：=q1^.next;
            while (q2<>nil) and (q2^.name<p^.name) do
              begin q1：=q2；q2：=q1^.next;   end;   查找结点 p 应插入的位置
            q1^.next：=p；p^.next：=q2；将 p 插入 q1 之后 q2 之前
          end;
  end;

procedure work;
  var i：integer;
       fn：string;
       f：text;
```

```
        p：point；
    begin
      write('Filename：')；readln(fn)；
      assign(f，fn)；reset(f)；
      readln(f，n)；
      for i：=1 to n do
        begin
          new(p)；p^.next：=nil；
          readln(f，p^.name)；
          insert(p)；
        end；
    end；

procedure print；  打印
    var p：point；
    begin
      p：=head^.next；
      while p<>nil do
        begin
          writeln(p^.name)；
          p：=p^.next；
        end；
    end；

begin
    clrscr；
    init；
    work；
    print；
end.
```

分析：(1)排序有多种方法，插入排序是其中的一种，其原理同摸扑克牌类似：摸到一张牌，把它按大小顺序插入牌中，每一张都如此处理，摸完后，得到的就是一副有序的扑克牌。本题可以用插入排序求解；

(2)为了减少移动数据的工作，可以采用链式存储结构。每个结点由两个域构成，数据域(用来存放姓名)和指针域(用来存放后继结点的地址)。如图 A 是将 1，3，6，7 按顺序构成一个线性链表的示意图。这样在这个有序表中插入一个 5 时，只需对指针进行相应地操作即可，如下图 B：

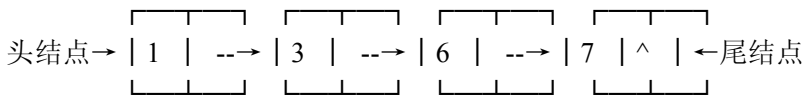


图 A

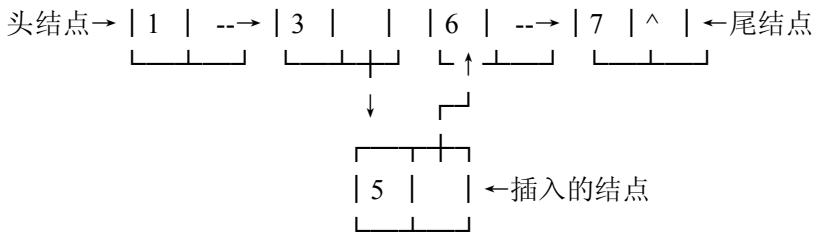


图 B

练习一

1、求 1987 乘幂的尾数：

M 和 N 是自然数，N>M>=1，而 1987^M 与 1987^N 的末三位数相同，求最小的 M 和 N。

分析：

(1) 本题只须记录 1987 的乘幂的末三位数，故不必高精度计算；

(2) 用数组 a[1..n]存储 1987 的 1 至 n 次幂的末三位数；

(3) n 的初始值为 2，计算 1987 的 n 次幂的末三位数，并和 1987 的 1 至 n-1 次幂进行比较，若无相等的，则 n=n+1，重复(3)；否则，第一次找到相等的，即是所求的 m，n 值。

2、一个特殊的数列：

写出两个 1，然后在它们中间插入 2 成为 121，下一步是在任意两个相邻的和数为 4 的数之间插入 3，成为 13231；再下一步又在任意两个相邻的和数为 4 的数之间插入 4，成为 1432341，...，由键盘输入 N(1<=N<=9)，求出用上面方法构造出来的序列，其最后插入的数为 N。

分析：字符串也可以看做是一个特殊的线性表，本题初始串是 11，对应 N=1 时的情况；然后在串中寻找相应的位置，依次插入 2，3，...，K。

3、求序列的第 300 项：

把所有 3 的方幂及互不相等的 3 的方幂和排列成一个递增序列：1，3，4，9，10，12，13，...，求这个序列的第 300 项。

分析：本题可以用一个线性表来记录这个递增的序列，通过递推可以将整个序列构造出来。方法如下：

(1)数组 a 存放线性表，t 为尾指针，b 存放 3 的幂，初始时 t=1，b=1；

(2)将 b 放入表尾，尾指针加 1；a[t]←b；t←t+1；

(3)将 b 依次与 1 至 t-1 的元素相加，按顺序放入表尾；

(4)重复(2)，(3)，直至第 300 项放入表中。

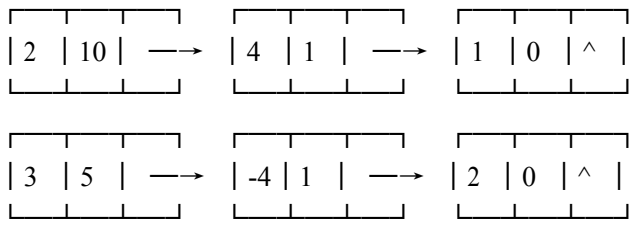
4、约瑟夫环(Joseph)

编号为 1，2，...，N 的 N 个人按顺时针方向围成一圈，每人持有一个密码(正整数)。一开始任选一个正整数作为报数上限值 M，从第一个人开始按顺时针方向自 1 开始报数，报到 M 时停止，报 M 的人出列，将他的密码作为新的 M 值，从他在顺时针方向上的第一个人开始重新从 1 报数，如此下去，直至所有有人全部出列为止。试设计一个程序求出列的顺序。

分析：这是一个数学游戏。N 个人围成一圈，依次报数，可以用一个循环链表模拟这一过程 。将链表的表尾指向表头，即形成循环链表。从某个人开始报数，报到 M 的人出列，也就是在在循环链表中删除相应的结点，然后依次删除完所有的结点，此时链表为空，头指针与尾指针相等。在具体操作中，要注意删除头结点和尾结点时指针的处理，谨防出错。

5、多项式的加法：试编程完成两个一元多项式的加法。

分析：大家都知道，两个一元多项式相加实际上就是合并同类项，最后结果一般要求按字母的升幂或降幂排列，比如：(2X^10+4X+1)+(3X^5-4X+2)=2X^10+3X^5+3。那么一元多项式在计算机中如何表示呢?为了节省空间，一个元多项式一般用一个线性表表示，线性表的每一个结点包括两个域：一个用来存放系数，一个用来存放指数，比如，上面相加的两个一元多项式可以分别表示为：

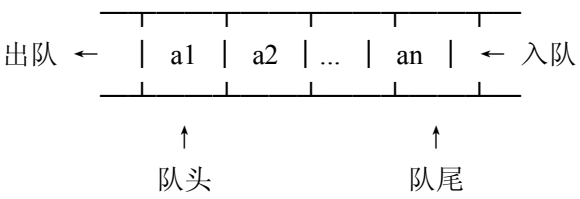


这样两个一元多项式相加就可以看成是归并两个链表。

第二节　队列

在日常生活中有许多“队列“的例子，如车站售票口买票的队伍，排在前面的人先买到票离开队伍，后来的人则加入队伍的末尾等候买票；其特点是“先进先出”(First In First Out)或“后进后出”(Last In Last Out)。

“队列”是在一端插入，另一端删除的特殊的线性表。进行删除的一端称为“队首”，进行插入的一端称为“队尾”(如下图)；插入也叫入队，删除则叫出队；在对队列进行操作时，一定要注意一头一尾。



例 1、有 N 张牌，记为 1，2，...，N，应当怎样排放，才能使：打开第一张是 1，然后报两张依次放在末尾；打开上面一张，刚好是 2，再依次打开上面一张，刚好是 3；如此继续下去，直至打开最后一张是 N。写一个程序解决这个问题。

解：Pascal 程序：

```
Program lt7_2_1；
uses crt；
var a，b：array[1..1000] of integer；
    i，j，t，h，n：integer；
begin
  clrscr；
  write('N=')；readln(n)；
  for i：=1 to n do a[i]：=i；
  a[1]：=1；h：=2；t：=n；b[1]：=1；
  for i：=2 to n do
```

```
    begin
      for j：=1 to i do
        begin
          inc(t)；a[t]：=a[h]；inc(h)；
        end；
      b[a[h]]：=i；inc(h)；
    end；
  for i：=1 to n do
    write(b[i]：5)；
end.
```

分析：这是一个典型队列的例子，请大家仔细体会在队列操作过程中头指针和尾指针的变化 。

例 2、集合的前 N 个元素：编一个程序，按递增次序生成集合 M 的最小的 N 个数，M 的定义如下：

(1)数 1 属于 M；

(2)如果 X 属于 M，则 Y=2*X+1 和 Z=3*1 也属于 M；

(3)此外再没有别的数属于 M。

解：Pascal 程序：

```
Program lt7_2_1；
uses crt；
var a，b：array[1..1000] of integer；
    x，ha，hb，t，total，n：integer；
begin
  clrscr；
  write('N=')；readln(n)；
  x：=1；a[1]：=1；
  ha：=1；hb：=1；t：=0；total：=1；
  while total<=n do
    begin
      write(x：5)；
      inc(t)；
      a[t]：=2*x+1；b[t]：=3*x+1；
      if a[ha]>b[hb] then begin
          x：=b[hb]；inc(hb)；
      end
      else begin
        x：=a[ha]；
        if a[ha]=b[hb] then inc(hb)；
        inc(ha)；
      end；
      inc(total)；
    end；
```

end.

分析：可以用两个队列来存放 Y 和 Z 中的数，分别用数组 a 和数组 b 存放。然后递推求出第 N 项，方法如下：

(1)令 ha 和 hb 分别为队列 a 和队列 b 的头指针，它们的尾指针为 t。初始时，X=1，ha=hb=t=1；

(2)将 2*x+1 和 3*x+1 分别放入队列 a 和队列 b 的队尾，尾指针加 1。即：

a[t]←2*x+1，b[t]←3*x+1，t←t+1；

(3)将队列 a 和队列 b 的头结点进行比较，可能有三种情况：

(A)a[ha]>b[hb]

(B)a[ha]=b[hb]

(C)a[ha]<b[hb]

将比较的小者取出送入 X，取出数的队列的头指针相应加 1。

(4)重复(2)，(3)直至取出第 N 项为止。

注意：数组的上标定到 1000，当 N 较大时会出现队满溢出的情况，可以将上标定大些，或改用循环链表进行改进。

练习二

1、高精度加法：

设计一个程序实现两个正整数(长度不超过 200 位)的求和运算。

解：Pascal 程序：

```
Program lx7_2_1；
uses crt；
var a，b：string；
    c，i，x，y，lm，ha，hb：integer；
    m：array[1..300] of integer；
begin
  clrscr；
  write('A=')；readln(a)；
  write('B=')；readln(b)；
  ha：=length(a)；hb：=length(b)；
  c：=0；lm：=1；
  while (ha>0) or (hb>0) do
    begin
      if ha>0 then x：=ord(a[ha])-48
          else x：=0；
      if hb>0 then y：=ord(b[hb])-48
          else y：=0；
      m[lm]：=x+y+c；
      c：=m[lm] div 10；
      m[lm]：=m[lm] mod 10；
      inc(lm)；dec(ha)；dec(hb)；
    end；
```

```
  if c>0 then    m[lm]：=c
      else dec(lm)；
  write(a，'+'，b，'=')；
  for i：=lm downto 1 do
     write(m[i])；
end.
```

2、基数排序：

将 278，109，063，930，589，184，505，269，008，083 利用基数排序法按从小到大的顺序排列。

分析：基数排序法是一种基于对关键字进行分配和收集的排序法，常用最低位优先法(Least Significant Digit first)，简称 LSD 法。它先从最低位的关键字开始进行排序，然后对次低位关键字进行排序，依此类推，直到对最高位进行排序为止。例如本题，关键字是各位上的数码，从 0 到 9 共 10 个。首先，将需要排序的数放在队列 A 中，如图：

3、有 1 至 2N 的自然数，按从小到大的顺序排成一列，对这 2N 个数进行如下操作：

（1）将这 2N 个数等分成 A，B 两组，即：

A 组：1，2，...，N； B 组：N+1，N+2，...，2N

（2）轮流从 A，B 两组中按顺序取数：

1，N+1，2，N+1，...，N，2N

（3）再将取过的数等分为两组，并重复上述操作，直到这 2N 个数又按从小到大的顺序排好为止。

例如：当 N=3 时，操作如下：

初始序列为：1，2，3，4，5，6

(1) 1，4，2，5，3，6

(2) 1，5，4，3，2，6

(3) 1，3，5，2，4，6

(4) 1，2，3，4，5，6

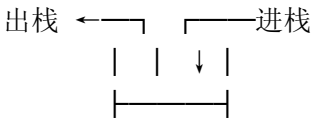
分析：将 1 至 2N 的自然数分成两组，用两个队列来模拟上述过程即可。

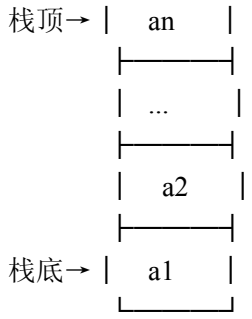
4、有一个数，它的末位数字是 N，将 N 移到这个数的首位，得到的新数恰好是原数的 N 倍，现输入 N 的值，求满足条件的最小的数。

第三节　栈

“栈”是一种先进后出(First In Last Out)或后进先出(Last In First Out)的数据结构。日常生活中也常能见到它的实例，如压入弹夹的子弹，最先压进去的子弹最后射出，而最后压入的子弹则最先发射出来。

“栈”是一种只能在一端进行插入和删除的特殊的线性表，进行插入和删除的一端称为“栈顶"，而不动的一端称为栈底(如下图)。插入的操作也称为进栈(PUSH)，删除的操作也称为出栈(POP)。





例 1、算术表达式的处理：由键盘输入一个算术表达式(含有+，-，*，/，(，)运算)，且运算结果为整数)，编一程序求该算术表达式的值。

分析：表达式的运算是程序设计中一个基本的问题，利用栈求解是一种简单易行的方法，下面介绍的是算符优先法。

比如：4+2*3-10/5

我们都知道，四则运算的法则是：(1)先乘除后加减，同级运算从左到右；(2)有括号先算括号。

因此上例的结果是 8。

算符优先法需要两个栈：一个是操作数栈，用来存放操作数，记为 SN；另一个是操作符栈用来存放运算符，记为 SP。具体处理方法如下：

(1)将 SN，SP 置为空栈；

(2)从左开始扫描表达式，若是操作数压入 SN 中；若是操作符则与 SP 的栈顶操作符比较优先级有两种可能：

（a）优先级小于栈顶算符，此时从 SN 中弹出两个操作数，从 SP 弹出一个操作符，实施运算，结果压入 SN 的栈顶。

（b）优先级大于栈顶算符，此时将操作符压入 SP 中。

(3)重复操作(2)直至表达式扫描完毕，这时 SP 应为空栈，而 SN 只有一个操作数，即为最后的结果。

为了方便起见，可以将#作为表达式的结束标志，初始化时在 SP 的栈底压入#，并将其优先级规定为最低。

下面给出的是计算 4+2*3-10/5#的示意图

步骤	SN	SP	读入字符	说明
1	空	#	4	将 4 压入 SN
2	4	#	+	将+压入 SP
3	4	#+	2	将 2 压入 SN
4	4 2	#+	*	将*压入 SP
5	4 2	#+*	3	将 3 压入 SN
6	4 2 3	#+*	-	-的优先级小于*，因此将 SN 中的 3，2 弹出，将 SP 中的*弹出，将 2*3 的结果压入 SN 中
7	4 6	#+	-	-的优先级小于其左边的+，因此将 SN 中的 4，6 弹出，将 SP 中的+弹出，将 4+6 的结果压入 SN 中
8	10	#	-	-压入 SP 中

```
 9      10        # -       10          | 10 压入 SN 中
10      10 10     # -       /           | 将/压入 SP 中
11      10 10     # - /     5           | 将 5 压入 SN 中
12      10 10 5 # - /     #             | #优先级小于/，故 SN 中的 10，5 弹出，SP 中
                                        | 的/弹出，将 10/5 的结果压入 SN 中
13      10 2      # -       #           | #优先级小于-，故 SN 中的 10，2 弹出，SP 中
                                        | 的-弹出，将 10-2 的结果压入 SN 中
14      8         #         #           | #与#相遇，运算结束，SN 中的 8 是最后计算
                                        | 的结果
```

解：Pascal 程序：

```
Program lt7_3_1；
uses crt；
const number：set of char=['0'..'9']；
        op：set of char=['+'，'-'，'*'，'/'，'('，')']；

var expr：string；
      sp：array[1..100] of char；
      sn：array[1..100] of integer；
      t，tp，n，tn：integer；

function can_cal(ch：char)：boolean；
  begin
    if (ch='#') or (ch=')') or ((sp[tp] in ['*'，'/']) and (ch in ['+'，'-']))
        then can_cal：=true else can_cal：=false；
  end；

procedure cal；
  begin
    case sp[tp] of
      '+'：sn[tn-1]：=sn[tn-1]+sn[tn]；
      '-'：sn[tn-1]：=sn[tn-1]-sn[tn]；
      '*'：sn[tn-1]：=sn[tn-1]*sn[tn]；
      '/'：sn[tn-1]：=sn[tn-1] div sn[tn]；
    end；
    dec(tn)；dec(tp)；
  end；

begin
  clrscr；
  write('Expression ：  ')；readln(expr)；
  write(expr+'=')；
  expr：=expr+'#'；
```

```
  tn：=0；tp：=1；sp[1]：='#'；t：=1；
  repeat
    if expr[t] in number then
       begin
         n：=0；
         repeat
           n：=n*10+ord(expr[t])-48；
           inc(t)；
         until not (expr[t] in number)；
         inc(tn)；sn[tn]：=n；
       end
    else begin
      if (expr[t]='(') or not can_cal(expr[t]) then
         begin
           inc(tp)；sp[tp]：=expr[t]；inc(t)；
         end
      else if expr[t]=')' then
        begin
          while sp[tp]<>'(' do cal；
          dec(tp)；inc(t)；
        end
        else cal；
    end；
  until (expr[t]='#') and (sp[tp]='#')；
  writeln(sn[1])；
end.
```

练习三

1、假设一个算术表达式中可包含三种括号：圆括号“(”和“)"；方括号“[”和“]”以及花括号“{”和“}"，且这三种括号可按任意的次序嵌套使用，试利用栈的运算，判别给定的表达式中所含括号是否正确配对出现。

分析：如果括号只有一种，比如说是“(”和“)"，则判断是否正确匹配可以这样进行：

用一个计数器 T 来记录左括号与右括号比较的情况，初始时 T=0，表达式从左向右扫描，如果遇到左括号则 T←T+1，遇到右括号则 T←T-1，中间如果出现 T<0 的情况，说明右括号的个数大于左括号的个数，匹配出错；扫描结束时，如果 T<>0，说明左右括号的个数不相等，匹配出错。

但本题的括号有三种，这里用栈可以较好地解决，方法如下：

(1)将“(”，“[”，“{”定为左括号，“)”，“]”，“}”定为右括号，建一个栈来存放左括号，初始时，栈为空；

(2)从左向右扫描表达式，如果是左括号，则将其压入栈中；如果是右括号则

（a）栈顶是与该右括号匹配的左括号，则将它们消去，重复(2)

（b）栈顶的左括号与该右括号不匹配，则匹配错误；

(3)扫描结束时，若栈非空，则匹配错误。

2、计算四个数：由键盘输入正整数 A，B，C，D(1<=A，B，C，D<=10)；然后按如下形式排列：

A□B□C□D=24；在四个数字之间的□处填上加，减，乘，除四种运算之一(不含括号)，输出所有满足条件的计算式子。若不能满足条件，则打印“NORESULT!“。

分析：A，B，C，D 四个数的位置已经固定，因此只需将+，-，*，/四种运算符依次放入三个□中，穷举所有可能的情况，再计算表达式的值即可。

第四节　数组

数组是大家非常熟悉的，我们可以把它看成是一个长度固定的线性表。本节在研究数组的一些处理方法的同时，还将介绍一些特殊的数组的存储和处理。

例一、矩阵填数：给出 N*N 的矩阵，要求用程序填入下列形式的数：

```
25 24 23 22 21
20 19 18 17 16
15 14 13 12 11
10  9  8  7  6
 5  4  3  2  1
```

解：Pascal 程序：

```
Program lt7_4_1；
uses crt；
const max=10；
var a：array[1..max，1..max] of integer；
    n：integer；

procedure work1；
  var i，j：integer；
  begin
    for i：=n downto 1 do
      for j：=n downto 1 do
        a[i，j]：=(n-i)*n+(n-j+1)；
  end；

procedure print；
  var i，j：integer；
  begin
    writeln；
    for i：=1 to n do
      begin
        for j：=1 to n do
          write(a[i，j]：5)；
```

```
            writeln；
          end；
    end；

begin
    clrscr；
    write('N=')；readln(n)；
    work；
    print；
end.
```

例二、在一个矩阵(N*N)中，若上三角中的元素全部为零，如下图所示：
为了节省空间，可用一个一维数组来表示这个矩阵。如右图　　1 0 0
可表示成为：(1 2 3 3 0 4)，在此种方法下，编程完成两个　　2 3 0
矩阵的加法。　　3 0 4

```
Program lt7_4_2；
uses crt；
const max=1000；
var a，b，c：array[1..max] of integer；
      n：integer；

procedure read_data；
    var i：integer；
    begin
      write('N=')；readln(n)；
      write('A ：  ')；
      for i：=1 to n*(n+1) div 2 do
        read(a[i])；
      write('B ：  ')；
      for i：=1 to n*(n+1) div 2 do
        read(b[i])；
    end；

procedure add；
    var i：integer；
    begin
      for i：=1 to n*(n+1) div 2 do
        c[i]：=a[i]+b[i]；
    end；

procedure print；
    var i，j，t：integer；
```

```
  begin
    t：=1；
    for i：=1 to n do
      begin
        for j：=1 to n do
          if j>i then write('0'：5)
              else begin write(c[t]：5)；inc(t)；end；
        writeln；
      end；
  end；

begin
  clrscr；
  read_data；
  add；print；
  writeln('Press any key to exit...')；
  repeat until keypressed；
end.
```

分析：本题应弄清楚下三角形矩阵中压缩存储及矩阵的加法运算法则。

练习四

1、数组的鞍点：已知数组 X(M，N)中的所有元素为互不相等的整数，编一个程序，先从每一行元素中找出最小的数，然后再从这些最小的数中找出最大的数。打印出这最大的数和它在数组中的下标。

2、稀疏矩阵：所谓的稀疏矩阵是指矩阵中的非 0 元素很少的一种矩阵，对于这种矩阵，可以用一个三元组表来存储非 0 元素，从而达到压缩存储，节省空间的目的。具体方法如下：假设 v 是稀疏矩阵 A 中第 i 行第 j 列的一个非 0 元素，那么该非 0 元素可以用一个三元组(i，j，v)表示；请编一个程序，从文件中读入一个用三元组表表示的稀疏矩阵，然后输出该稀疏矩阵的转置矩阵。(文件第一行为：M，N，K，分别表示稀疏矩阵的行和列以及非 0 元素的个数，以下 K+1 行是 K 个三元组)

3、蛇形填数：给一个 N*N 矩阵按下列方式填数。

```
 1  3  4 10 11
 2  5  9 12 19
 6  8 13 18 20
 7 14 17 21 24
15 16 22 23 25
```

第八章 搜索

搜索是人工智能的基本问题。在程序设计中，许多问题的求解都需要利用到搜索技术，它是利用计算机解题的一个重要手段。

问题的状态可以用图来表示，而问题的求解则往往是从状态图中寻找某个状态，或是寻找从一个状态到另一个状态的路径。这一求解的过程可能并不象解一个一元二次方程那样有现成的方法，它需要逐步探索与总是问题有关的各种状态，这即是搜索。

本章将介绍广度优先搜索和深度优先搜索及其递归程序的实现。

第一节 深度优先搜索

所谓"深度"是对产生问题的状态结点而言的，"深度优先"是一种控制结点扩展的策略，这种策略是优先扩展深度大的结点，把状态向纵深发展。深度优先搜索也叫做 DFS 法(Depth First Search)。

例一、设有一个 4*4 的棋盘，用四个棋子布到格子中，要求满足以下条件:

(1)任意两个棋子不在同一行和同一列上;

(2)任意两个棋子不在同一条对角线上。

试问有多少种棋局，编程把它们全部打印出来。

解：PASCAL 程序：

```
Program lt9_1_1;
uses crt;
const n=4;
var a:array[1..n] of integer;
    total:integer;

function pass(x，y:integer):boolean;
  var i，j:integer;
  begin
    pass:=true;
    for i:=1 to x-1 do
      if (a[i]=y) or (abs(i-x)=abs(a[i]-y)) then
          begin pass:=false;exit;end;
  end;

procedure print;
  var i，j:integer;
  begin
    inc(total);
```

```
    writeln('[', total, ']');
    for i:=1 to n do
      begin
        for j:=1 to n do
          if j=a[i] then write('O ')
            else write('* ');
        writeln;
      end;
  end;

procedure try(k:integer);
  var i:integer;
  begin
    for i:=1 to n do
      if pass(k, i) then
        begin
          a[k]:=i;
          if k=n then print
            else try(k+1);
          a[k]:=0;
        end;
  end;

begin
  clrscr;
  fillchar(a, sizeof(a), 0);
  total:=0;
  try(1);
end.
```

分析:这里要求找出所有满足条件的棋局，因此需要穷举所有可能的布子方案，可以按如下方法递归产生:

```
    令 D 为深度，与棋盘的行相对应，初始时 D=1;
    Procedure try(d:integer);
    begin
      for i:=1 to 4 do
        if 第 i 个格子满足条件 then
          begin
            往第 d 行第 i 列的格子放入一枚棋子;
            如果 d=4 则得一方案，打印
            否则试探下一行，即 try(d+1);
            恢复第 d 行第 i 列的格子递归前的状态;
          end;
```

```
    end;
```

这种方法是某一行放入棋子后，再试探下一行，将问题向纵深发展;若本行试探完毕则回到上一行换另一种方案。这样必定可穷举完所有可能的状态。从本题可以看出，前面所说的递归回溯法即体现了深度优先搜索的思想。上面对深度优先算法的描述就是回溯法常见的模式。

例二、在 6*6 的方格中，放入 24 个相同的小球，每格放一个，要求每行每列都有 4 个小球(不考虑对角线)，编程输出所有方案。

解:Pascal 程序:

```
Program lx9_1_2;
uses crt;
const n=6;
var map:array[1..n，1..n] of boolean;
    a:array[1..n] of integer;
    total:longint;

procedure print;
  var i，j:integer;
  begin
    inc(total);gotoxy(1，3);
    writeln('['，total，']');
    for i:=1 to n do
      begin
        for j:=1 to n do
          if map[i，j] then write('* ')
              else write('O ');
        writeln;
      end;
  end;

procedure try(k:integer);
  var i，j:integer;
  begin
    for i:=1 to n-1 do
      if a[i]<2 then begin
          map[k，i]:=true;
          inc(a[i]);
          for j:=i+1 to n do
            if a[j]<2 then begin
                map[k，j]:=true;
                inc(a[j]);
                if k=n then print
```

```
                    else try(k+1);
                map[k，j]:=false;
                dec(a[j]);
              end;
            map[k，i]:=false;
            dec(a[i]);
        end;
  end;

begin
  clrscr;
  fillchar(map，sizeof(map)，false);
  fillchar(a，sizeof(a)，0);
  try(1);
end.
```

分析:本题实际上是例一的变形;

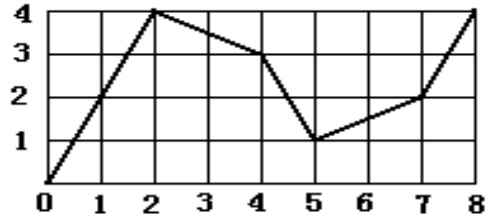
(1)把枚举每行每列四个小球转化成为每行每列填入 2 个空格;

(2)用两重循环实现往一行中放入两个空格;

(3)用数组 B 记录搜索过程中每列上空格的个数;

(4)本题利用深度搜索求解时要注意及时回溯，以提高效率，同时要注意退出递归时全局变量的正确恢复。

例三、跳马问题:在半张中国象棋盘上，有一匹马自左下角往右上角跳，今规定只许往右跳，不许往左跳，图(A)给出的就是一种跳行路线。编程计算共有多少种不同的跳行路线，并将路线打印出来。



解：PASCAL 程序：

```
Program lt9_1_2;
uses crt;
const d:array[1..4，1..2] of shortint=((2，1)，(1，2)，(-1，2)，(-2，1));
var a:array[1..10，1..2] of shortint;
      total:integer;

function pass(x，y，i:integer):boolean;
  begin
    if (x+d[i，1]<0) or (x+d[i，1]>4) or (y+d[i，2]>8)
       then pass:=false else pass:=true;
```

```
  end;

procedure print(k:integer);
  var i:integer;
  begin
    inc(total);
    write('[，total，'] : (0，0)');
    for i:=1 to k do
      write('->(，a[i，1]，'，'，a[i，2]，')');
    writeln;
  end;

procedure try(x，y，k:integer);
  var i:integer;
  begin
    for i:=1 to 4 do
      if pass(x，y，i) then
          begin
            a[k，1]:=x+d[i，1];a[k，2]:=y+d[i，2];
            if (a[k，1]=4) and (a[k，2]=8) then print(k)
                else try(a[k，1]，a[k，2]，k+1);
          end;
  end;

begin
  clrscr;
  total:=0;
  try(0，0，1);
  writeln('Press any key to exit..。');
  repeat until keypressed;
end.
```

分析:(1)这里可以把深度 d 定为马跳行的步数，马的位置可以用它所在的行与列表示因此初始时马的位置是(0，0);

(2)位置在(x，y)上的马可能四种跳行的方向，如图(B)，这四种方向，可以按 x，y 的增量分别记为(2，1)，(1，2)，(-1，2)，(-2，1)

(3)一种可行的跳法是指落下的位置应在棋盘中。

练习一:

1、有一括号列 S 由 N 个左括号和 N 个右括号构成，现定义好括号列如下:

(1) 若 A 是好括号列，则(A)也是;

(2) 若 A 和 B 是好括号列，则 AB 也是好的。

例如:(()(()))是好的，而(()))(()则不是，现由键盘输入 N，求满足条件的所的好括号列，并

打印出来。
解:Pacal 程序:

```
Program lx9_1_1;
uses crt;
var n:integer;
      total:longint;

procedure try(x，y:integer;s:string);
   var i:integer;
   begin
      if (x=n) and (y=n) then begin
          inc(total);writeln('['，total1，'] '，s);
      end
      else begin
         if x<n then try(x+1，y，s+'(');
         if y<x then try(x，y+1，s+')');
      end;
   end;

begin
   clrscr;
   write('N=');readln(n);
   total:=0;try(0，0，'');
end.
```

分析:从好括号列的定义可知，所谓的"好括号列"就是我们在表达式里所说的正确匹配的括号列，其特点是:从任意的一个位置之前的右括号的个数不能超过左括号的个数。由这个特点，可以构造一个产生好括号列的方法:用 x，y 记录某一状态中左右括号的个数;若左括号的个数小于 N(即 x<N)，则可加入一个左括号;若右括号的个数小于左括号的个数，则可加入一个右括号，如此重复操作，直至产生一个好括号列。
2、排列组合问题：从数码 1-9 中任选 N 个不同的数作不重复的排列（或组合），求出所有排列（或组合）的方案及总数。
3、迷宫问题：在一个迷宫中寻找从入口（最左上角的格子）到出口（最右下角的格子）的路径。
4、填数游戏一:以下列方式向 5*5 的矩阵中填入数字。设数字 i(1<=i<=25)已被置于座标位置(x，y)，则数字 i+1 的座标位置应为(z，w)，(z，w)可根据下列关系由(x，y)算出。
(1) (z，w)=(x±3，y)
(2) (z，w)=(x，y±3)
(3) (z，w)=(x±2，y±2)
例如数字 1 的起始位置座标被定为(2，2)，则数字 2 的可能位置座标是：(5，2)，(2，5)，或(4，4)。编写一个程序，当数字 1 被指定于某一起始位置时，列举其它 24 个数字应在的位置，列举出该条件下的所有可有的方案。
解:同例二类似，只不过方向增量变为(3，0)，(-3，0)，(0，3)，(0，-3)，(2，2)，(2，-2)，(-2，2)，(-2，-2)，

Pascal 程序:

```
Program lx9_1_3;
uses crt;
const n=5;
        d:array[1..8，1..2] of shortint=((3，0)，(-3，0)，(0，3)，(0，-3)，
                                                    (2，2)，(2，-2)，(-2，2)，(-2，-2));
var x0，y0:byte;
      a:array[1..n，1..n] of byte;
      total:longint;

procedure print;
   var i，j:integer;
   begin
      inc(total);
      gotoxy(1，3);
      writeln('['，total，']');
      for i:=1 to n do
         begin
            for j:=1 to n do
               write(a[i，j]:3);
            writeln;
         end;
   end;

procedure try(x，y，k:byte);
   var i，x1，y1:integer;
   begin
      for i:=1 to 8 do
         begin
            x1:=x+d[i，1];y1:=y+d[i，2];
            if (x1>0) and (y1>0) and (x1<=n)
                and (y1<=n) and (a[x1，y1]=0) then
                   begin
                      a[x1，y1]:=k;
                      if k=n*n then print
                           else try(x1，y1，k+1);
                      a[x1，y1]:=0;
                   end;
         end;
   end;

begin
```

```
  clrscr;
  write('x0，y0=');readln(x0，y0);
  fillchar(a，sizeof(a)，0);
  total:=0;a[x0，y0]:=1;
  try(x0，y0，2);
  writeln('Total='，total);
  writeln('Press any key to exit..。');
  repeat until keypressed;
end.
```

5、填数游戏二。有一个 M*N 的矩阵，要求将 1 至 M*N 的自然数填入矩阵中，满足下列条件:

(1)同一行中，右边的数字比左边的数字大;

(2)同一列中，下面的数字比上面的数字大。

打印所有的填法，并统计总数。

解:Pascal 程序:

```
$Q-，R-，S-
Program lx9_1_4;
uses crt;
const m=3;n=6;
var a:array[0..m，0..n] of integer;
      used:array[1..m*n] of boolean;
      total:longint;

procedure print;
  var i，j:integer;
  begin
    inc(total);gotoxy(1，3);
    writeln('['，total，']');
    for i:=1 to m do
      begin
        for j:=1 to n do
          write(a[i，j]:3);
        writeln;
      end;
  end;

procedure try(x，y:integer);
  var i:integer;
  begin
    for i:=x*y to m*n-(m-x+1)*(n-y+1)+1 do
      if not used[i] and (i>a[x-1，y]) and (i>a[x，y-1]) then
          begin
```

```
            a[x，y]:=i;used[i]:=true;
            if i=m*n-1 then print
                else begin
                   if y=n then try(x+1，1)
                        else try(x，y+1);
                end;
            used[i]:=false;
        end;
end;

begin
   clrscr;
   fillchar(used，sizeof(used)，false);
   fillchar(a，sizeof(a)，0);
   a[1，1]:=1;a[m，n]:=m*n;
   used[1]:=true;used[m*n]:=true;
   try(1，2);
   writeln('Total='，total);
end.
```

分析:本题可以将放入格子中的数字的个数作为深度，先往格子(1，1)放第一个数，然后依次往格子(1，2)，(1，3)，...，(m，n-1)，(m，n)填数字，每填一个数时应如何判断该数是否满足条件，做到及时回溯，以提高搜索的效率是非常关键的。为此需要认真研究题目的特点。根据题意可以知道:在任何一个 K*L 的格子里，最左上角的数字必定是最小的，而最右下角的数字必定是最大的，故有:

(1)格子(1，1)必定是填数 1。格子(m，n)必定填数 m*n;

(2)若 A 是格子(x，y)所要填入数，则有:x*y<=A<=m*n-(m-x+1)*(n-y+1)+1;

6、反幻方:在 3*3 的方格中填入 1 至 9，使得横，竖，对角上的数字之和都不相等。下图给出的是一例。请编程找出所有可能的方案。

1	2	3
4	5	8
6	9	7

图 一

分析:

(1)深度优先搜索。用一个二维数组 A 来存储这个 3*3 的矩阵。

(2)用 x 表示行，y 表示列，搜索时应注意判断放入格子(x，y)的数码是否符合要求;

（a）如果 y=3，就计算这一行的数码和，其值存放在 A[x，4]中，如果该和已出现过，则回溯;

（b）如果 x=3，则计算这一列的数码和，其值存放在 A[4，y]中，并进行判断是否需要

回溯;

（c）如果 x=3，y=1 还应计算从左下至右上的对角线的数码和;

（d）如果 x=3，y=3 还应计算从左上至右下的对角线的数码和。

为了提高搜索速度，可以求出本质不同的解，其余的解可以由这些本质不同的解通过旋转和翻转得到。为了产生非本质解，搜索时做如下规定:

（a）要求 a[1，1]<a[3，1]<a[1，3]<a[3，3];

（b）要求 a[2，1]>a[1，2].

解：略

7、将 M*N 个 0 和 1 填入一个 M*N 的矩阵中，形成一个数表 A，

```
        | a11   a12 ...   a1n  |
    A=  | a21   a22 ...   a2n  |
        | .....                |
        | am1 am2 ... amn  |
```

数表 A 中第 i 行和数的和记为 ri(i=1，2，...，m)，它们叫做 A 的行和向量，数表 A 第 j 列的数的和记为 qj(j=1，2，...，m)，它们叫做 A 的列和向量。现由文件读入数表 A 的行和列，以及行和向量与列和向量，编程求出满足条件的所的数表 A。

分析:本题是将例题一一般化，将若干个 1 放入一个 M*N 的方阵中，使得每行和每列上的 1 的个数满足所给出的要求。

思路:(1)应该容易判断，若 r1+r2+...+rn<>q1+q2+...+qm，则问题无解;

(2)将放入 1 的个数看做是深度，1 的位置记为(x，y)，其中 x 代表行，y 代表列，第一个 1 应从(1，1)开始试探;

(3)往 k 行放入一个 1 时，若前一个 1 的位置是(k，y)，则它的位置应在第 k 行的 y+1 列至(m-本行还应放入 1 的个数+1)这个范围内进行试探;若这一列上已放入 1 的个数小于 qy，则该格子内放入一个 1，并记录下来;否则换一个位置试探。

```
Program lx9_1_6;
uses crt;
const max=20;
var m，n，s1，s2:integer;
    map:array[1..max，1..max] of 0..1;
    a，b，c，d:array[1..max] of integer;
    total:longint;

procedure error;
  begin
    writeln('NO ANSWER!');
    writeln('Press any key to exit..。');
    repeat until keypressed;
    halt;
  end;

procedure init;
  var f:text;
      fn:string;
```

```
        i，j:integer;
    begin
      write('Filename:');readln(fn);
      assign(f，fn);reset(f);
      readln(f，m，n);s1:=0;s2:=0;
      for i:=1 to m do
        begin read(f，a[i]);s1:=s1+a[i];end;
      for i:=1 to n do
        begin read(f，b[i]);s2:=s2+b[i];end;
      close(f);
      if s1<>s2 then error;
      fillchar(map，sizeof(map)，0);
      fillchar(c，sizeof(c)，0);
      fillchar(d，sizeof(d)，0);
    end;

procedure print;
    var i，j:integer;
    begin
      inc(total);gotoxy(1，3);
      writeln('['，total，']');
      for i:=1 to m do
        begin
          for j:=1 to n do
            write(map[i，j]:3);
          writeln;
        end;
    end;

procedure try(x，y，t:integer);
    var i，j:integer;
    begin
      for i:=y+1 to n-(a[x]-c[x])+1 do
        if (map[x，i]=0) and (d[i]<b[i]) then
            begin
              map[x，i]:=1;inc(c[x]);inc(d[i]);
              if t=s1 then print
                  else if (x<=m) then begin
                     if c[x]=a[x] then try(x+1，0，t+1)
                          else try(x，i，t+1);
                  end;
              map[x，i]:=0;dec(c[x]);dec(d[i]);
```

```
            end;
  end;

begin
  clrscr;
  init;
  try(1，0，1);
  if total=0 then writeln('NO ANSWER!');
end.
```

第二节　广度优先搜索

广度优先是另一种控制结点扩展的策略，这种策略优先扩展深度小的结点，把问题的状态向横向发展。广度优先搜索法也叫 BFS 法(Breadth First Search)，进行广度优先搜索时需要利用到队列这一数据结构。

例一、分油问题:假设有 3 个油瓶，容量分别为 10，7，3(斤)。开始时 10 斤油瓶是满的，另外两个是空的，请用这三个油瓶将 10 斤油平分成相等的两部分。

解：PASCAL 程序：

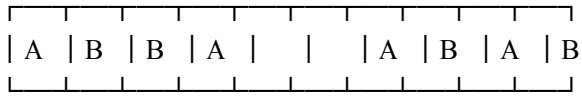
```
Program lt9_2_2;
uses crt;
const max=3000;
        v:array[1..3] of byte=(10，7，3);

type node=record
          a:array[1..3] of byte;
          ft:integer;
       end;
var o:array[1..max] of node;
     h，t，i，j，k:integer;

function is_ans(h:integer):boolean;
  begin
    with o[h] do
      if ord(a[1]=5)+ord(a[2]=5)+ord(a[3]=5)=2
           then is_ans:=true else is_ans:=false;
  end;

function new_node(t:integer):boolean;
  var r:integer;
  begin
    r:=t;
```

```
    repeat
      dec(r);
      with o[t] do
        if (a[1]=o[r].a[1]) and (a[2]=o[r].a[2]) and (a[3]=o[r].a[3])
          then begin new_node:=false;exit;end;
    until (r=1);
    new_node:=true;
  end;

procedure print(r:integer);
  var b:array[1..30] of integer;
      t，l，p:integer;
  begin
    t:=0;
    while r>0 do
      begin
        inc(t);b[t]:=r;
        r:=o[r].ft;
      end;
    gotoxy(1，1);
    writeln(':5，v[1]:5，v[2]:5，v[3]:5);
    writeln('--------------------');
    for l:=t downto 1 do
      begin
        write('[，t-l:2，'] ');
        for p:=1 to 3 do
          write(o[b[l]].a[p]:5);
        writeln;
      end;
    halt;
  end;

begin
  clrscr;
  with o[1] do
    begin
      a[1]:=10;a[2]:=0;a[3]:=0;
    end;
  h:=1;t:=2;
  repeat
    if is_ans(h) then print(h);
    for i:=1 to 3 do
```

```
        if o[h].a[i]>0 then
          for j:=1 to 3 do
            if (i<>j) and (o[h].a[j]<v[j]) then
              begin
                for k:=1 to 3 do
                  o[t].a[k]:=o[h].a[k];
                with o[t] do
                  begin
                    ft:=h;
                    if o[h].a[i]>(v[j]-a[j]) then
                      begin
                        a[i]:=o[h].a[i]+a[j]-v[j];a[j]:=v[j];
                      end
                    else begin
                      a[j]:=a[j]+o[h].a[i];a[i]:=0;
                    end;
                  end;
                if new_node(t) then inc(t);
              end;
      inc(h);
    until (h>t);
    writeln('NO ANSWER!');
end.
```

例二、中国盒子问题:给定 2*N 个盒子排成一行，其中有 N-1 个棋子 A 和 N-1 个棋子 B，余下是两个连续的空格，

如下图，是 N=5 的一种布局。



移子规则为:任意两个相邻的棋子可移到空格中，且这两个棋子的次序保持不变。

目标:全部棋子 A 移到棋子 B 的左边。

练习二:

1、跳棋:跳棋的原始状态如下图(A)，目标状态如下图(B)，其中 0 代表空格，W 代表白子，B 代表黑子，跳棋的规则是:(1)任一个棋子可以移到空格中去;(2)任一个棋子可以跳过 1 个或两个棋子移到空格中去。试编一个程序，用最少的步数将原始状态移成目标状态。

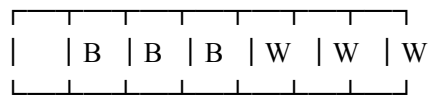


图 A

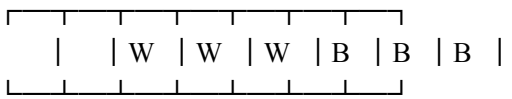
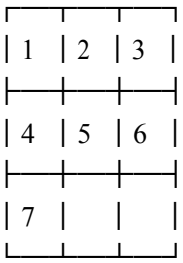


图 B

2、七数码问题:

在 3*3 的棋盘上放有 7 个棋子，编号分别为 1 到 7，余下两个是空格。与空格相邻的一个棋子可以移到空格中，每移动一次算一步，现任给一个初始状态，要求用最少的步数移成下图所示的目标状态。



3、N 个钱币摆放一排，有的钱币正面朝上(记为 1)，有的钱币正面朝下(记为 0)，每次可以任意改变 K 个钱币的状态，即正反面互换。编一个程序判定能否在有限的步数内使得所有钱币正面朝上，若能，请给出步数最少的方案。

4、下图 A 所示的是一个棋盘，放有编号为 1 到 5 的 5 个棋子，如果两个格子中没有线分隔，就表示这两个格子是相通的，编一个程序，用最少的步数将图 A 的状态移成图 B 所示的状态(一次移动一子，无论多远算一步)。

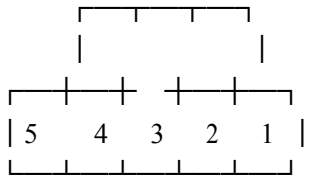


图 A

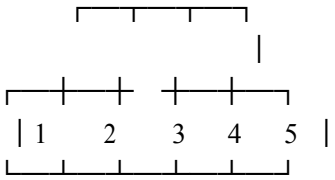


图 B

5、迷宫问题：在一个迷宫中找出从入口到出口的一条步数最少的通路。

第九章 其他常用知识和算法

本章将抛砖引玉地探讨一些程序设计中常用的知识及算法，包括图论及其基本应用，动态规划等。要想对这些问题进行深一步地研究，可以系统地学习图论、组合数学、运筹学等书籍。

第一节 图论及其基本算法

图是另一种有层次关系的非线性的数据结构。在日常生活中有图的许多实例，如铁路交通网，客运航空线示意图，化学结构式，比赛安排表等。下面 的如几个例子都可称为图。在实际运用中，我们可以用图来表示事物间相互的关系，从而根据需要灵活地构建数学模型。例如：图(A)，可以用点来表示人，如果两个人相互认识，则在表示这两个人的点之间连一条线。这样从图中我们就可以清楚地看到，这些人之间相互认识的关系。图(B)：可以用点表示城市，若两城市间有连线则表示可在这两城市架设通信线路，线旁的数字表示架设这条线路的费用。图(C)：4 个点表示 4 支足球队，它们进行循环比赛，若甲队胜乙队，则连一条由甲队指向乙队的有向线段。在上面三个例子中，(A)，(B)又可称为无向图，(C)称为有向图，其中(B)是一个有权图。

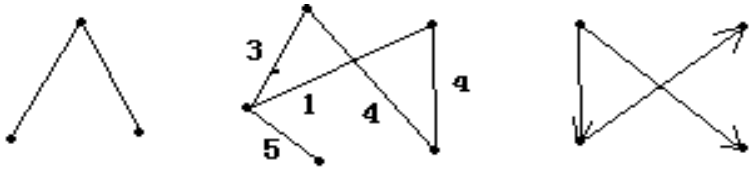
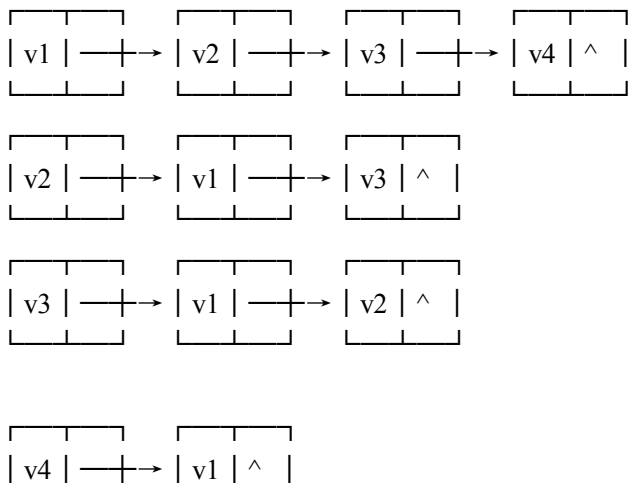


图 A　　图 B　　图 C

图常用的存储方式有两种，一种是邻接表法，另一种是邻接矩阵法。

邻接表表示图有两个部分：一部分表示某点的信息，另一部分表示与该点相连的点。例如：图 A 的邻接表如下所示：



邻接矩阵是用一个二维数组表示图的有关信息，比如图 A 的邻接矩阵为：

点	v1	v2	v3	v4
v1	0	1	1	1
v2	1	0	1	0
v3	1	1	0	0
v4	1	0	0	0

在邻接矩阵中，第 i 行第 j 表示图中点 i 和点 j 之间是否有连线，若有则值为 1，否则为0。比如说：点 v1 和 v2 之间有连线，则矩阵的第一行第二列的值为 1。而矩阵的第四行行三列的值为 0 说明图中点 v4 和 v3 之间没有连线。图 A 是一个无向图，它的邻接矩阵是一个关于主对角线对称的矩阵。而有向图的邻接矩阵则不一定是对称的。比如图 C 的邻接矩阵为:

点	v1	v2	v3	v4
v1	0	1	1	1
v2	0	0	1	1
v3	0	0	0	0
v4	0	0	1	0

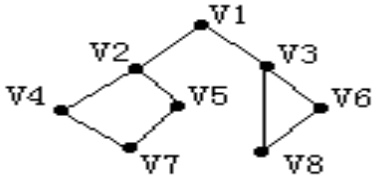
例一、图的遍历：请编一个程序对下图进行遍历。

分析："遍历"是指从图的某个点出发，沿着与之相连的边访问图中的每个一次且仅一次。基本方法有两种：深度优先遍历和广度优先遍历。

深度优先和广度优先遍历，与前面所说的树的深度与广度优先遍历是类似的：比下图中，如果从点 V1 出发，那么：

深度优先遍历各点的顺序为：v1，v2，v4，v7，v5，v3，v6，v8。

广度优先遍历各点的顺序为：v1，v2，v3，v4，v5，v6，v7，v8。



下面是两种方法的 Pascal 程序：

解：程序一：深度优先遍历

可以仿照前面树的深度优先遍的得出图的深度优先遍历的递归算法，这里是用栈来实现的非递归算法，算法如下：

(1)利用一个栈 S 来记录访问的路径，由于从点 1 出发，因此初始时 S[1]：=1;

(2)找一个与栈顶的点相连而又未被访问过的点，如找到，则输出并压入栈顶;如果未找到，则栈顶的点弹出。

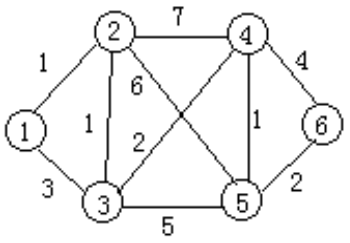
(3)重复(2)直到所有的点均已输出。如果栈指针为 0，而尚的点未输出，说明该图不是一个连通图。

```
Program lt8_1_a;
uses crt;
const n=8;
        v：array[1..8，1..8] of 0..1=((0，1，1，0，0，0，0，0)，
                                                (1，0，0，1，1，0，0，0)，
                                                (1，0，0，0，0，1，1，0)，
                                                (0，1，0，0，0，0，0，1)，
                                                (0，1，0，0，0，0，0，1)，
                                                (0，0，1，0，0，0，1，0)，
                                                (0，0，1，0，0，1，0，0)，
                                                (0，0，0，1，1，0，0，0));
var visited：array[1..n] of boolean;
      s：array[1..n] of byte;
      top，p，i，total：integer;
      find：boolean;
begin
   clrscr;
   fillchar(visited，sizeof(visited)，false);
   write('V1 ');s[1]：=1;visited[1]：=true;
   total：=1;top：=1;p：=1;
   repeat
      find：=false;
      for i：=1 to n do
         if not visited[i] and (v[p，i]=1) then
              begin
                 inc(top);s[top]：=i;p：=i;
                 visited[p]：=true;find：=true;
                 inc(total);write('V'，p，' ');break;
              end;
      if not find then
          begin p：=s[top];dec(top);end;
   until total=n;
   writeln;
end.
```

程序二：进行广度优先遍历(程序略)

例二、最短路径。

下图是一个铁路交通图的例子。图中的顶点表示车站，分别用 1，2，...，6 编号，两个点之间的连线则表示铁路线路，而线旁的数字则表示该条线路的长度。要求编一个程序，求出从车站 1 至车站 6 的最短路径。



解：本题是在一个有权图中求给定两点之间的最短路径，也许大家会考虑用搜索的方法试探所有可能经过的路线，再从中找出最小者，这样在理论上是成立的，但可能效率不高，这里介绍一种有效的算法，就是 Dijkstra 算法。

PASCAL 程序：

```
Program lt8_2;
uses crt;
const max=100;
var map：array[1..max，1..max] of integer;
    d：array[1..max，1..2] of integer;
    p：array[1..max] of boolean;
    n：integer;

procedure init;
  var i，j：integer;
      f：text;
      fn：string;
  begin
    write('Filename：');readln(fn);
    assign(f，fn);reset(f);
    readln(f，n);
    for i：=1 to n do
      for j：=1 to n do
        read(f，map[i，j]);
    close(f);
    fillchar(p，sizeof(p)，false);
    fillchar(d，sizeof(d)，0);
  end;

function find_min：integer;
  var i，min，t：integer;
  begin
    min：=maxint;
    for i：=1 to n do
      if not p[i] and (d[i，1]>0) and (d[i，1]<min) then
```

```
            begin
              min：=d[i，1];t：=i;
            end;
      find_min：=t;
   end;

procedure change(t：integer);
   var i：integer;
   begin
      for i：=1 to n do
         if not p[i] and (map[t，i]>0)
            and (d[i，1]=0) or (d[i，1]>d[t，1]+map[t，i]) then
              begin d[i，1]：=d[t，1]+map[t，i];d[i，2]：=t;end;
   end;

procedure dijkstra;
   var i，j，t：integer;
   begin
      p[1]：=true;t：=1;
      repeat
         change(t);
         t：=find_min;
         p[t]：=true;
      until p[n];
   end;

procedure print;
   var i，t：integer;
       r：array[1..max] of integer;
   begin
      t：=1;r[1]：=6;
      while d[r[t]，2]>0 do
         begin
           r[t+1]：=d[r[t]，2];inc(t);
         end;
      writeln('From V1 to V6');
      for i：=t downto 2 do
         write('V'，r[i]，'->');
      writeln('V'，r[1]);
      writeln('Min='，d[6，1]);
   end;
```

```
begin
  clrscr;
  init;
  dijkstra;
  print;
end.
```

练习一、

1、拓扑排序。

有 N 个士兵(1<=N<=26)，依次编号为 A，B，...，Z。队列训练时，指挥官要把士兵从高到矮依次排成一行，但现在指挥官不能直接获得每个人的身高信息，只能获得"P1 比 P2 高"这样的比较结果，记为"P1>P2"，如"A>B"表示 A 比 B 高。现从文本文件读入比较结果的信息，每个比较结果在文本文件中占一行。编程输出一种符合条件的排队方案。若输入的数据无解，则打印"NO ANSWER!"。

例如：若输入为：

A>B

B>D

F>D

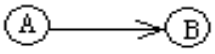
则一种正确的输出为：

ABFD

分析：

(1)由局部可比信息求得全局可比信息就是拓扑排序。

(2) 局部信息可以用一个有向图表示。如 A>B，则在加一条由点 A 指向点 B 的有向线。那么上例可以用右图表示出来。

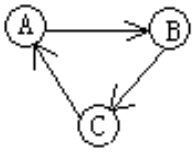


(3)拓扑排序的方法：

（a）从图中选一个无前驱的点输出，同时删去由该点出发的所有的有向线;

（b）重复 a，直至不能进行为止。此时若图中还有点未输出，则问题无解;否则，输出的序列即为一可行方案。

注意：在步骤(3)的 b 中，如果图中还有点未输出，说明图中有一个回路(即环)。这在拓扑排序中是矛盾的。比方说有如右图所示的一个环，其表示出来关于身高的信息是：A>B，B>C，C>A，这显然是矛盾的，因此无解。



解：略

2、 地图四色：用不超过四种的颜色给一个地图染色，使得相邻的两个地区所着的颜色各不相同。

3、最小部分树。

有一个城市分为 N 个区，现要在各区之间铺设有线电视线路。任意两个区之间铺设线路的费用如下图所示，其中图的顶点表示各个区，线旁的数字表示铺设该条线路的费用。要求设计一个费用最省的铺线方案使得每一个区均能收看到有线电视

分析：(1)这个问题实际上是求图的最小部分树。因为每一个区均能收看到有线电视，所以该图是连通的;又因为要求费用最省，所以该图无圈。

(2)求图的最小部分树常用 Prim 算法(也称加边法)和 Kurskal 算法(也称破圈法)

这里介绍的是 Prim 算法。具体操作如下：

（a）去掉图中所有的线，只留下顶点;

（b）从图中任意选出一个点加入集合 S1，余下的点划入集合 S2;

（c）从所有连接集合 S1 与 S2 的点的线中选一条最短的加入图中，并将该线所连的集合 S2 中的点从集合 S2 中删去，加入集合 S1 中;

（d）重复步骤 c，直至往图中加入 N-1 条边(N 为顶点的个数)。

4、一笔画：编一个程序对给定的图进行一笔画。

分析：图的一笔画是指从图的某一个顶点出发，经过图中所有的边一次且仅一次。现由文本文件读入一个图的邻接矩阵，判断该图能否一笔画，若能，则给出一笔画的方法。

分析：一笔画是图论中研究得比较早的一个问题，一个图能够一笔画的条件是：

(1)该图是一个连通图;

(2)该图至多有 2 个度为奇数的点。

这里所说的"度"是指与一个点相连的边的数目，如果边的条数是奇数，则该点的度是奇数，否则是偶数。

在一笔画问题中，如果连通图所有的点的度均为偶数，则一笔画时可以从任意一点出发，最后又能回到起点。如果有两个点的度为奇数，则一笔画时，一定是从一个奇数度的点出发，最后到达另一个奇数度的点。此外，在连通图中，奇数度的点总是成对出现的。一笔画所经过的路线又叫欧拉路(如果是回路的话，也叫欧拉回路)。

找欧拉路(或回路)可以用 Fleury 算法，如下：

(1)找一个出发点 P(如果图中有两个奇数度的点话，任取其一)。

(2)找一条与 P 相连的边，伸展欧拉路(选边的时候应注意，最后再选取断边;断边是：当去掉该边后使得图不连通的边)。

(3)将选出的边所连的另一个点取代 P，去掉该边，重复(2)，直至经过所有的边。

5、哈密尔顿问题。

哈密尔顿问题是指在一个图中找出这样的一条路：从一个图的顶点出发，经过图中所有顶点一次且仅一次。象这样的路称为哈密尔顿路。现由文本文件读入一个图的邻接表，判断该图是否有哈密尔顿路，若有则输出。

6、图的关节点。

如果从一个连通图中删去某点 V，使得该图不连通，那么 V 点就称为该图的一个关节点 。现从文本文件中读入一个图的邻接矩阵，试编程找出该图所有的关节点。

7、 有一集团公司下有 N 个子公司，各子公司由公路连接，现要在 N 个子公司中选一个出来成立总装车间，装配各子公司送来的部件，且使得各子公司到总装车间的路程总和最小。

第二节　动态规划

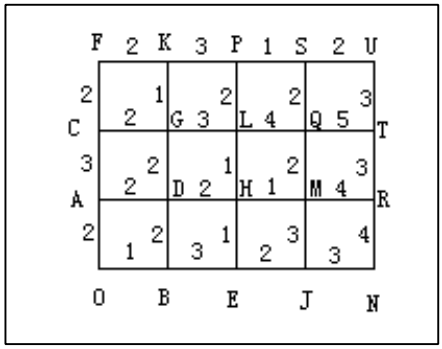
动态规划是近来发展较快的一种组合算法，是运筹学的一个分支，是解决多阶段决策过程最优化的一种数学方法。我们可以用它来解决最优路径问题，资源分配问题，生产调度问题，库存问题，装载问题，排序问题，设备更新问题，生产过程最优控制问题等等。

在生产和科学实验当中，有一类活动的过程，可将它分成若干个阶段，在它的每个阶段要作出决策，从而使全局达到最优。当各个阶段决策确定后，就组成一个决策序列，因而也就决定了整个过程的一条活动路线。这种把一个过程看作一个前后相关具有链状结构的多阶段过程就称为多阶段决策过程。

所谓动态是指在多阶段决策问题中，各个阶段采取的决策，一般来说是与时间有关的，决策依赖于当前的状态，又随即引起状态的转移，一个决策序列就是在变化的状态中产生，故有"动态"的含义。

下面，我们结合最短路径问题来介绍动态规划的基本思想。

求下图中点 O 到点 U 的最短距离(假设只许往上和往右走)。



从点 O 到点 U，可以按经过的路径，分成七个阶段，分别为：O->AB->CDE->FGHJ->KLMN->PQR->ST->U。

最短路径有一个重要特性：如果点 O 经过点 H 到达点 U 是一条最短路径，则在这条最优路径上由点 H 出发到达点 U 的子路径，是由点 H 出发到达点 U 所有可能选择的不同路径的最短路径(证明略)。根据这一特点，寻找最短路径的时候，可以从最后一段开始，用由后向前逐段递推的方法，求出个点到点 U 的最短路径。

如若考虑到从点 O 到点 U 的最短路径，也是该路径上个点到点的最短路径，令 O 点到 U 点的最短距离为 dO，A 点到 U 点的最短距离为 dA，...，故有：

dO=min{2+dA，1+dB}，

dA=min{3+dC，2+dD}，

dB=min{2+dD，3+dE}，

.....

dQ=min{5+dT，2+dS}.

下面按照动态规划的方法，将上例从最后一段开始计算，由后向前逐步递推移至 O 点。计算步骤如下：

阶段 7：从 S 点或 T 点到达 U 点，这时各自只有一种选择，故：

dS=2;dT=3;

阶段 6：出发点有 P，Q，R 三个，其中 Q 点到达 U 点有两种选择，或是经过 S 点，或是经过 T 点，故：

dQ=min{2+dS，5+dT}=min{4，8}=4;

dP=min{1+dS}=min{3}=3;

dR=min{3+dT}=min{6}=6;

阶段 5：出发点有 K，L，M，N 四个，同理有：

dK=min{3+dP}=min{6}=6;

dL=min{2+dP，4+dq}=min{5，8}=5;

dM=min{2+dQ，4+dR}=min{6，10}=6;

dN=min{4+dR}=min{10}=10;

阶段 4：出发点有 F，G，H，J 四个，同理有：

dF=min{2+dK}=min{8}=8;

dG=min{1+dK，3+dL}=min{7，8}=7;

dH=min{1+dL，1+d}=min{6，7}=6;

dJ=min{3+dM，3+dN}=min{9，13}=9;

阶段 3：出发点有 C，D，E 三个，同理有：

dC=min{2+dF，2+dG}=min{10，9}=9;

dD=min{4+dG，2+dH}=min{11，8}=8;

dE=min{1+dH，2+dJ}=min{7，11}=7;

阶段 2：出发点有 A，B 两个，同理有：

dA=min{3+dC，2+dD}=min{12，10}=10;

dB=min{2+dD，3+dE}=min{10，10}=10;

阶段 1：出发点是 O，同理有：

dO=min{2+dA，1+dB}=min{12，11}=11.

由此得到全过程的最短路径是 11，并且可以由以上推导过程反推得最短的路线是：O->B->D->H->L->P->S->U;或 O->B->E->H->L->P->S->U。

从上可以知道，能应用动态规划解决的问题，必须满足最优性原则，动态规划的关键在于正确的写出基本的递推关系式和恰当的边界条件。它需要把问题的过程化成几个互相联系的阶段，恰当的选取状态变量和决策变量及定义最优值函数，从而把一个大问题化成一族同类型的子问题，然后逐个求解。即从边界条件开始，逐段递推寻优，在每一个子问题的求解中，都利用到前面的子问题的最优化结果，依次进行，最后一个子问题的最优解，就是整个问题的最优解。

例一、数字三角形：下图给出了一个数字三角形.请编一个程序计算从顶至底的某处的一条路径，使得该路径所经过的数字的总和最大。对于路径规定如下：

(1)每一步可沿左斜线向下走或右斜线向下走;

(2)1<三角形的行数<=100;

(3)三角形中的数字为整数 0，1，...，99;

输入数据：由文件 INPUT.TXT 中首先读出的是三角形的行数。在例子中 INPUT.TXT 表示如下：

5

7

⑦

③ 8

⑧ 1 0

2 ⑦ 4 4

4 ⑤ 2 6 5

3 8
8 1 0
2 7 4 4
4 5 2 6 5

输出最大的总和。如上例为：30(其路径如图圈的数字)。

分析：(1)假设最优路径(即总和最大)为⑦→③→⑧→⑦→⑤，则子路径⑦→③→⑧→⑦必定是从初始点⑦到中间点⑦的所有路径中最优的，否则，如果从初始点⑦到中间点⑦还有另一条的路径更优，假设为⑦→a1→a2→⑦，则新路径⑦→a1→ a2→⑦→⑤则优于原来假设的最优路径⑦→③→⑧→⑦→⑤，与假设矛盾。从以上反证法可以清楚地知道：数字三角形问题满足动态规划的最优性原则，可以利用动态规划求解。

(2)采用顺推法：记第 i 行第 j 列的数字为 a(i, j)，从初始点到 a(i, j)的最大总和记为 f(i, j);则应有第一层上的数字的 a(1，1)的最大总和为它本身，即：

f(1，1)=a(1，1);

以下各层按如下方法递推：

f(i，j)=a(i，j)+max{f(i-1，j-1)，f(i-1，j)}

(3)用以上方法递推计算完最后一层，最后一中寻最大值即为本题的解.

解：Pascal 程序：

```
Program lt10_2_1;
uses crt;
const max=100;
var n：integer;
    a：array[0..max，0..max] of longint;

procedure work;
  var f：text;
      i，j：integer;
  begin
    assign(f，'input.txt');
    reset(f);
    readln(f，n);
    fillchar(a，sizeof(a)，0);
    for i：=1 to n do
      for j：=1 to i do
        begin
          read(f，a[i，j]);
          if a[i-1，j-1]>a[i-1，j] then
              a[i，j]：=a[i，j]+a[i-1，j-1]
              else a[i，j]：=a[i，j]+a[i-1，j];
        end;
    close(f);
  end;
```

```
procedure print;
  var max：longint;
      i：integer;
  begin
    max：=0;
    for i：=1 to n do
      if a[n，i]>max then max：=a[n，i];
    writeln('Max='，max);
  end;

begin
  clrscr;
  work;print;
end.
```

习题二：

1、 某工业生产部门根据国家计划的安排，拟将某种高效率的五台机器，分配给所属的 A，B，C 三个工厂，各工厂若获得这种机器后，可以为国家盈利如下表，问：这五台机器如何分配给各工厂，才能使国家盈利最大?

单位：万元

P	A	B	C
0	0	0	0
1	3	5	4
2	7	10	6
3	9	11	11
4	12	11	12
5	13	11	12

其中：p 为盈利，s 为机器台数。

2、已知：f(x)=x1^2+2x2^2+x3^2-2x1-4x2-2x3

x1+x2+x3=3 且 x1，x2，x3 均为非负整数。

求 f(x)的最小值。

3、N 块银币中有一块不合格，不合格的银币较正常为重，现用一天平找出不合格的来，要求最坏情况不用天平的次数最少。

4、某一印刷厂有 6 项加工任务，对印刷车间和装订车间所需时间见下表：

任　务	J1	J2	J3	J4	J5	J6
印刷车间	3	12	5	2	9	11
装订车间	8	10	9	6	3	1

时间单位：天

完成每项任务都要先去印刷车间印刷，再到装订车间装订。问怎样安排这 6 项加工任务的加工工序，使得加工总工时最少。

5、有一个由数字 1，2，...，9 组成的数字串(长度不超过 200)，问如何 M(1<=M<=20) 个加号插入这个数字串中，使得所形成的算术表达式的值最小。

注意：(1)加号不能加在数字串的最前面或最末尾，也不应有两个或两个以上的加号相邻;

(2)M 保证小于数字串的长度。

例如：数字串 79846，若需加入两个加号，则最佳方案是 79+8+46，算术表达式的值是 133。

输入格式：从键盘读入输入文件名。数字串在输入文件的第一行行首(数字串中间无空格且不换行)，M 的值在输入文件的第二行行首。

输出格式：在屏幕上输出最小的和。

备注： Pascal 字符与符号

1. 标识符

(1) 标识符的定义：标识符就是以字母开头的字母数字序列，有效长度为 63 个字符，并且大小写等效。可以用来标示常量、变量、程序、函数等。例如例 1.1 中的 Area(程序名)，pi(符号常量)，s、r(变量名)都是标识符。

(2) 标识符的分类：

a.保留字(关键字)

所谓保留字是指在 Pascal 语言中具有特定的含义，你必须了解它的含义，以便于正确的使用，否则会造成错误。标准 Pascal 语言中的保留字一共有 35 个，Turbo Pascal 语言一共有 51 个。下面是 Pascal 语言的保留字:

AND，ARRAY，BEGIN，CASE，CONST，DIV，DO，DOWNTO，ELSE，END，FILE，FOR，FUNTION，GOTO，IF，IN，LABEL，MOD，NIL，NOT，OF，OR，PACKED，PROCEDURE，PROGRAM，RECORD，REPEAT，SET，THEN，TO，TYPE，UNTIL，VAR，WHILE，WITH 等

b.标准标识符：指 Pascal 语言预先定义的标识符，具有特殊含义。

以下列举了 Turbo Pascal 语言部分常用的标准表识符：

标准常量	False	Maxint	True				
标准类型	Boolean	Char	Real	Integer			
标准函数	Abs	Arctan	Chr	Cos	Eof	Eoln	Exp
	Ln	Odd	Ord	Pred	Round	Sin	Sqr
	Sqrt	Succ	Trunc				
标准过程	Dispose	Get	New	Pack	Page	Put	Read
	Readln	Reset	Rewrite	Unpack	Write	Writeln	
标准文件	Input	Output					

c. 用户自定义标识符：由你自己根据需要来定义。

（1）选用的标识符不能和保留字相同。

（2）语法上允许预定义的标准标识符作为你自己定义的标识符使用，但最好还是不要用。

以下列举了你自己在定义标识符时可以用的字符：

A——Z; a——z; 0——9; +, -, *, /, =, <>, <=, >=, <, >, (,), [,], {, }, :=, ,, ;, ., :, .., ', ^

Pascal 数据类型

数据是程序设计的一个重要内容，其重要特征----数据类型，确定了该数据的形、取值范围以及所能参与的运算。

Turbo Pascal 提供了丰富的数据类型，这 些数据类型可以分为三大类：简单类型、构造

类型和指针类型，其中简单类型可以分为标准类型（整型、实型、字符型和布尔型）和自定义类型（枚举型和子界型），构造类型可以分为数组类型、集合类型、记录类型和文件类型。这些数据类型中除了指针类型是动态数据类型外，其他的都是静态数据类型。在这些数据类型中的简单类型都是有序类型，除了实型以外的简单类型都是顺序类型，所谓顺序类型就是他们的值不仅是有序的而且是有顺序号。

在这里主要介绍整型、实型、字符型和布尔型四种常用的数据类型。

1.整型

一个整型数据用来存放整数。Turbo Pascal 支持五种预定义整型，它们是 shortint（短整型）、 integer（整型）、 longint（长整型）、 byte（字节型）和 word（字类型），Turbo Pascal 分别用相同的名字作为他们的标识符。每一种类型规定了相应的整数取值范围以及所占用的内存字节数。

类型	数值范围	占字节数	格式
shortint	-128..128	1	带符号 8 位
inteter	-32768..32767	2	带符号 16 位
longint	-2147483648..2147483647	4	带符号 32 位
byte	0..255	1	带符号 8 位
word	0..65535	2	带符号 16 位

Turbo Pascal 规定了两个预定义整型常量标识符 maxint 和 maxlonint，他们各表示确定的常数值，maxint 为 32767， longint 为 2147483647，他们的类型分别是 integer 和 longint

2.实型

一个实型数据用来存放实数。Turbo Pascal 支持五种预定义实型，它们是 real（基本实型）、single（但精度实型）、double（双精度实型）、extended（扩展实型）、comp（装配实型），Turbo Pascal 分别用相同的名字作为他们的标识符。每一种类型规定了相应的实数取值范围、所占用的内存字节数以及它们所能达到的精度

类型	数值范围	占字节数	有效位数
real	2.9e-39..1.7e38	6	11..12
single	1.5e-45..3.4e38	4	7..8
double	5.0e-324..1.7e308	8	15..16

Turbo Pascal 支持两种用于执行实型运算的代码生成模式：软件仿真模式和 80x87 浮点模式。除了 real 可以在软件仿真模式下直接运行以外，其他类型必须在 80x87 浮点模式下运行。

3.布尔型

一个布尔型数据用来存放逻辑值（布尔值）。布尔型的值只有两个：false 和 true，并且 false 的序号是 0，true 的序号是 1。false 和 true 都是预定义常数标识符，分别表示逻辑假和逻辑真。并且 true<false。boolean 是布尔型的标识符。

4.字符型

字符型用 char 作为标识符。字符型必须用单引号括起来，字母作为字符型时，大小写是不等价的，并且字符型只允许单引号中有一个字符，否则就是字符串。

常量与变量

1.常量

(1)常量：在某个程序的整个过程中其值不变的量。

(2)常量定义：常量定义出现在说明部分。它的语法格式是：

const
<常量标识符>=<常量>;
...
<常量标识符>=<常量>;

常量标识符的类型由定义它的常量的类型决定。例如:const a=12 隐含说明 a 是整型;const r=3.21 隐含说明 r 是实型.....

(3)常量定义部分必须以保留字 const 开头，可以包含一个或几个常量定义，而且每个常量均以分号结束。

(4)Turbo Pascal 类型常量

类型常量，又称变量常数，它是 Turbo Pascal 的一个扩充特性。类型常量的定义与标准 Pascal 规定的常数定义和变量说明有所区别。类型常量定义的语法格式：

const
<简单类型常量标识符>:简单类型=常数;

例如：

```
const
counter:integer=0;
flag:boolean=true;
index:0..100=0;
```

2.变量

(1)变量：在某个程序中的运行过程中其值可以发生改变的量

(2)变量说明：变量说明出现在说明部分。它的语法格式是：

var
<变量标识符列表>:<类型>;
...
<变量标识符列表>:<类型>;

其中，保留字 var 表示开始一个变量说明部分。变量标识符列表是一个用逗号隔开的标识符序列，冒号后面的类型是类型标识符。每个变量说明均以分号结束。

例如：

```
var
a,b,c:integer;
m,n:real;
```

标准函数

1.算术函数

函数标识符	自变量类型	意义	结果类型
abs	整型、实型	绝对值	同自变量
arctan	整型、实型	反正切	实型
cos	整型、实型	余弦	实型
exp	整型、实型	指数	实型
frac	整型、实型	小数部分	实型
int	整型、实型	整数部分	实型
ln	整型、实型	自然对数	实型

pi	无自变量	圆周率	实型
sin	整型、实型	正弦	实型
sqr	整型、实型	平方	同自变量
sqrt	整型、实型	平方根	实型

例：abs(-4)=4　　abs(-7.49)=7.49　　arctan(0)=0.0
　　sin(pi)=0.0　　cos(pi)=-1.0　　frac(-3.71)=-0.71
　　int(-3.71)=-3.0　　sqr(4)=16　　sqrt(4)=2

2.标准函数

函数标识符	自变量类型	意义	结果类型
odd	整型	判断奇数	布尔型
pred	离散类型	求前趋	同自变量
succ	离散类型	求后继	同自变量

例：odd(1000)=false　　pred(2000)=1999　　succ(2000)=2001
　　odd(3)=true　　pred('x')='w　　succ('x')='y'

3.转换函数

函数标识符	自变量类型	意义	结果类型
chr	byte	自变量对应的字符	字符型
ord	离散类型	自变量对应的序号	longint
round	实型	四舍五入	longint
trunc	实型	截断取整	longint

例：chr(66)='B'　ord('A')=65　round(-4.3)=-5　　trunc(2.88)=2

4.杂类函数

函数标识符	自变量类型	意义	结果类型
random	无自变量	[0,1 间的随机实数	real
random	word	[0,自变量间的随机整数)	word
randomize	无自变量	初始化内部随机数产生器	longint
upcase	字符型	使小写英文字母变为大写	字符型
downcase	字符型	使小写英文字母变为大写	字符型

运算符和表达式

1.运算符和优先级

(1)运算符

是实型，如果全部的运算对象都是整型并且运算不是除法，则结果为整型，若运算是除法，则结果是实型　　a.算术运算符

运算符	运算	运算对象	结果类型
+	加	整型、实型	只要有一个运算对象是实型，结果就
-	减	整型、实型	是实型，如果全部的运算对象都是整
*	乘	整型、实型	型并且运算不是除法，则结果为整型，
/	除	整型、实型	若运算是除法，则结果是实型。

div	整除	整型	整型
mod	取余	整型	整型

b.逻辑运算符

运算符	运算	运算对象	结果类型
not	逻辑非	布尔型	布尔型
and	逻辑与	布尔型	布尔型
or	逻辑或	布尔型	布尔型
xor	逻辑异或	布尔型	布尔型

c.关系运算符

运算符	运算	运算对象	结果类型
=	等于	简单类型	布尔型
<>	不等于	简单类型	布尔型
<	小于	简单类型	布尔型
>	大于	简单类型	布尔型
<=	小于等于	简单类型	布尔型
>=	大于等于	简单类型	布尔型

(2)优先级

运算符	优先级
not	1(高)
*,/,div,mod,and	2
xor,+,-,or	3
in,=,<>,>=,<=,<>	4(低)

2.表达式

(1)算术表达式：算术表达式是由算术运算符连接常量、变量、函数的式子。算术表达式中各个运算符的次序为： ( )-->函数-->*,/,div,mod-->+,1

(2)布尔表达式：Turbo Pascal 提供给布尔表达式以下基本操作：逻辑运算和关系运算。

（3）数学上的表达式与 pascal 语言表达式的区别

数学表达式	PASCAL 表达式	注意
2a	2*a	*号不能省略
a÷b	a/b	除号的写法
a≠b	a<>b	不等号的写法
a≤b	a<=b	小于等于号的写法

思考与练习：

1、熟记 Pascal 的保留字和标准标识符，明确自定义标识符的定义要点。

2、取整函数 int 与截断取整函数 trunc 有什么区别？举例说明。

3、判断以下标识符的合法性：

a3　　3a　　a17　　abcd　　ex9.5　　α　　β　　λ

5、将下列的数学表达式改写成 PASCAL 表达式：

b^2-4ac

6、求下列表达式的值：

20 mod 19　　15 mod 9　　7 div 8　　19 div 3

(4>5)　and　(7<8)

(8>9)　or (9<10)

2　and　((3=3)　or　(3<7))