第三章 FORTRAN 结构化程序设计

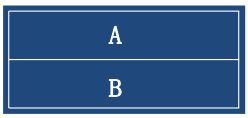
结构化程序设计

按照一定的结构形式来设计和编写程序，以便阅读与检查。其包括顺序结构、选择结构、循环结构（当型循环或直到型循环）。

3.1 顺序结构程序设计

概述

上后下，先左后右；即先执行 A，再执行 B，不存在跳转与循环。



案例分析

输入 3 个气象站 5 个月（汛期）雨量数据，统计每个气象站的总雨量和平均雨量，计算 3 个站五月、六月、七月、八月、九月的平均雨量，输出每个气象站每个月的雨量、总雨量和平均雨量，以及五月、六月、七月、八月、九月的平均雨量。

站名	汛期各月雨量（毫米）				
	5月	6月	7月	8月	9月
江阴	76.8	176.5	308.1	41	69.6
定波闸	71.5	208.5	352.1	47.2	62.6
肖山	65.5	200	239.7	44.3	63

```
PROGRAM ex06_02
IMPLICIT NONE
REAL r11,r12,r13,r14,r15,total11,av11
REAL r21,r22,r23,r24,r25,total21,av21
REAL r31,r32,r33,r34,r35,total31,av31
REAL av1,av2,av3,av4,av5
WRITE(*,“(28X,‘5 月  6 月  7 月  8 月  9 月’)”)
WRITE(*,“(1X,‘输入江阴气象站五个月的雨:’,\)”)          READ(*,*) r11,r12,r13,r14,r15
WRITE(*,“(1X,‘输入定波闸气象站五个月的雨量:’,\)”)      READ(*,*) r21,r22,r23,r24,r25
WRITE(*,“(1X,‘输入肖山气象站五个月的雨量:’,\)”)        READ(*,*) r31,r32,r33,r34,r35
FORMAT(F5.1,F5.1,F5.1,F5.1,F5.1)
total11=r11+r12+r13+r14+r15              av11=total11/5
total21=r21+r22+r23+r24+r25              av21=total21/5
total31=r31+r32+r33+r34+r35              av31=total31/5
av1=(r11+r21+r31)/3        av2=(r12+r22+r32)/3        av3=(r13+r23+r33)/3
av4=(r14+r24+r34)/3        av5=(r15+r25+r35)/3
WRITE(*,“(26X,‘5 月  6 月  7 月  8 月  9 月  总雨量     平均雨量’)”)
WRITE(*,200)‘江阴气象站五个月的雨量:’, r11,r12,r13,r14,r15,total11,av11
WRITE(*,200)‘定波闸气象站五个月的雨量:’, r21,r22,r23,r24,r25,total21,av21
WRITE(*,200)‘肖山气象站五个月的雨量:’, r31,r32,r33,r34,r35,total31,av31
200 FORMAT(1X,A22,5(F5.1,2X),F6.1,2X,F7.3)
WRITE(*,300) ‘5 月’,av1,‘6 月’,av2,‘7 月’,av3,‘8 月’,av4,‘9 月’,av5
300   FORMAT(1X,A4,‘平均雨量：’,F7.3)
END
```

3.2 选择结构程序设计

3.2.1 选择结构简介

选择结构　依据给定的条件做逻辑判断，再根据判断的结果决定应执行哪种操作。

案例

（1）输入学生成绩，判定合格与否，输出判定结果。
（2）已知三个整数 A,B,C,输入其值并打印三个数中最大值。
（3）暴雨预警信号分三级，分别以黄色、橙色、红色表示。

3.2.2 IF 语句

总体概述　Fortran 提供了 3 种典型的块 IF 结构：① 单分支　② 双分支　③ 多分支

3.2.2.1 单分支块 IF 结构

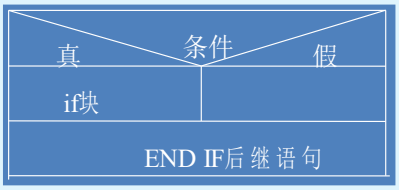
一般形式

```
IF(条件) THEN
    IF 块
END IF
```

① 块 IF 语句“IF（条件）THEN”。它是块 IF 结构的入口语句。
② IF 块。它是一个语句序列，由若干条可执行语句组成。
③ END IF 语句。它是块 IF 结构的出口语句。

案例分析：从键盘输入一个气温值，如果大于 35.0，则显示在屏幕上。

```
PROGRAM ex04_02
IMPLICIT NONE
REAL T
READ *,T
IF(T>35.0) THEN
    PRINT *, '这是高温'
    PRINT *,   'T=',T
END IF
END
```

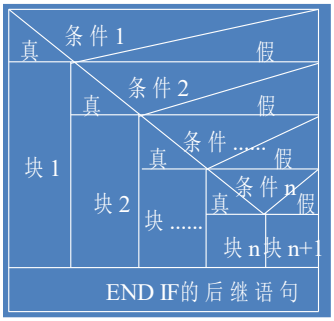


3.2.2.2 双分支块与多分支块 IF 结构

一般形式

```
IF（条件 1） THEN
    块 1
ELSE IF（条件n） THEN
    块 n
[ ELSE
    块 n + 1 ]
END IF
```

每一个其他条件后面都有 THEN



案例分析

由于大气受到污染，一些地区开始形成酸雨区，酸雨是指 PH 值小于 5.6 的雨雪或其他形式的大气降水。通过收集水样测量其 PH 值，判断它的酸碱性并打印出来。

```
PROGRAM ex07_03;        IMPLICIT NONE ;      REAL ph
WRITE(*,*)'Please enter PH value:';
READ *,ph
IF(ph<5.6) THEN
    WRITE(*,100) ph
ELSE
    WRITE(*,200) ph
END IF
100 FORMAT(1X,'PH=',F4.2,',is acid rain!');
200 FORMAT(1X,'PH=',F4.2,',is not acid rain!')
END
```

案例

在气象部门发布的天气预报中小雨、中雨、暴雨等专业术语，它们之间的区别如表所示：

降雨强度	降雨量（12小时，单位：mm）	降雨量（24小时，单位：mm）
小雨	<5	<10
中雨	5～15	10～25
大雨	15～30	25～50
暴雨	30～70	50～100
大暴雨	70～140	100～250
特大暴雨	>140	>250

```
PROGRAM ex07_04
IMPLICIT NONE
REAL r
WRITE(*,100)
READ *,r
IF(r<5) THEN
    PRINT  200
ELSE IF(r<15) THEN
    PRINT  300
ELSE IF(r<30) THEN
    PRINT  400
ELSE IF(r<70) THEN
    PRINT  500
ELSE IF(r<140) THEN
    PRINT  600
ELSE
    PRINT  700
END IF
100 FORMAT(1X,'请输入 12 小时降雨量：')
200 FORMAT(1X,'小雨')
300 FORMAT(1X,'中雨')
400 FORMAT(1X,'大雨')
500 FORMAT(1X,'暴雨')
600 FORMAT(1X,'大暴雨')
700 FORMAT(1X,'特大暴雨')
END
```

3.2.2.4 逻辑 IF 结构

一般形式　**IF（条件） 语句**

条件可以是一个合法的逻辑表达式或关系表达式，语句是一个合法的可执行语句，且只有一条语句。

案例

根据层结参数N^2的大小可以判断大气层结状态：大气层结状态 $=\begin{cases}\text{稳定层结} & N^2>0\\ \text{中性层结} & N^2=0\\ \text{不稳定层结} & N^2<0\end{cases}$

```
PROGRAM ex07_05;   IMPLICIT NONE;      REAL N2
WRITE(*,100);      READ(*,*)   N2
IF(N2.EQ.0.0) PRINT *,'中性层结'          等于
IF(N2.GT.0.0) PRINT *,'稳定层结'          大于
IF(N2.LT.0.0) PRINT *,'不稳定层结'        小于
100 FORMAT(1X,'PLEASE INPUT N2:')
END
```

3.2.3 SELECT CASE 语句

描述　CASE 结构是一种多路分支选择结构，可有多个分支可供选择。其实质是判断选择表达式的值是否与某一控制表达式的值相匹配。

说明　选择表达式和控制表达式可以为整型、逻辑型或字符型（没有实型）。控制表达式可以是一个不重复的值或一组同类值的列表，如：

① 用逗号分隔的单个值列表。如：

CASE('a','b','c','x','y','z')，当选择表达式的值为 a,b,c,x,y,z 之一时，执行相应的语句块。

CASE(3,6,9)，当选择表达式的值为 3，6 或 9 时，执行相应的语句块。

② 用冒号分隔的值的范围。如：

一般形式

```
SELECT CASE（选择表达式）
    CASE（控制表达式 1）
        块 1
    CASE（控制表达式 2）
        块 2
    CASE（控制表达式 n）
        块 n
    [CASE DEFAULT
        默认块]
END SELECT
```

CASE('a':'g')，当选择表达式的值落入 a～g 范围内时，执行相应的语句块。
CASE(5:10)，当选择表达式的值落入 5～10 范围内时，执行相应的语句块。
CASE(10:)，当选择表达式的值大于或等于 10 时，执行相应的语句块。
CASE(:10)，当选择表达式的值小于或等于 10 时，执行相应的语句块。

案例

根据风对地上物体所引起的现象将风的大小分为 13 个等级，称为风力等级，以 0～12 等级数字记载，如下表所示：现对所输入的风速进行分类并输出。

等级	名称	风速	陆地物象	海面波浪	浪高
0	无风	0.0 ~ 0.2	烟直上	平静	0
1	软风	0.3 ~ 1.5	烟示风向	微波峰无飞沫	0.1
2	轻风	1.6 ~ 3.3	感觉有风	小波峰未破碎	0.2
3	微风	3.4 ~ 5.4	旌旗展开	小波峰顶破裂	0.6
4	和风	5.5 ~ 7.9	吹起尘土	小浪白沫波峰	1
5	劲风	8.0 ~ 10.7	小数摇摆	中浪折沫峰群	2
6	强风	10.8 ~ 13.8	电线有声	大浪到个飞沫	3
7	疾风	13.9 ~ 17.1	步行困难	破峰白沫成条	4
8	大风	17.2 ~ 20.7	折毁树枝	浪长高有浪花	5.5
9	烈风	20.8 ~ 24.4	小损房屋	浪峰倒卷	7
10	狂风	24.5 ~ 28.4	拔起树木	海浪翻滚咆哮	9
11	暴风	28.5 ~ 32.6	损毁普遍	波峰全呈飞沫	11.5
12	台风/飓风	32.7以上	摧毁巨大	海浪滔天	14

```
PROGRAM ex07_06
IMPLICIT NONE
REAL wind_velocity
PRINT *,'请输入风速：'
READ *,wind_velocity          乘十取整
SELECT CASE(INT(wind_velocity*10))
    CASE(0:2)                 case 语句不支持实型
        PRINT *,'0 级'
        PRINT *,'无风'
    CASE(3:15)
        PRINT *,'1 级'
        PRINT *,'软风'
    CASE(16:33)
        PRINT *,'2 级'
        PRINT *,'轻风'
    （此处代码省略）
    CASE(285:326)
        PRINT *,'11 级'
        PRINT *,'暴风'
    CASE(327:)
        PRINT *,'12 级'
        PRINT *,'台风/飓风'
    CASE DEFAULT
        PRINT *,'非法数据'
END SELECT
END
```

3.2.4 选择语句的嵌套

描述

在一个块 IF 结构中都可以完整地包含一个（或多个）块 IF 结构，即构成块 IF 的嵌套结构

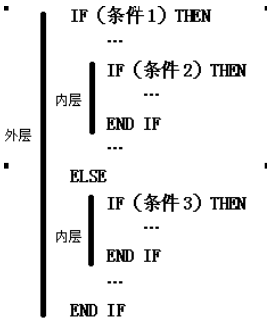
特殊注意

当嵌套层次过多时，往往一时难以找到同一层的块 IF 中的各语句，一般可按以下方法确定：

① 从最内层的块 IF 语句开始，向下找到离它最近的 END IF 语句，将它们用线括起来，这就是同一层次的块 IF。

② 由内向外重复这一个过程，直到遇见最外层块 IF 语句和 END IF 语句为止。

③ 在书写嵌套分支结构时采取缩进方式进行程序书写，程序的嵌套层次就容易确定。



案例-求解当系数 *a,b,c* 为不同情况下的一元二次方程根

```
READ *,a,b,c
d=b**2-4.0*a*c
IF(a==0.0) THEN
    IF(b==0.0) THEN
        IF(c==0.0) THEN
            PRINT *,'平凡解'
        ELSE
            PRINT *,'无解'
        END IF
```

```
        ELSE
            PRINT *,'一个实根'
            PRINT *,-c/d
        END IF
    ELSE
        IF(d>0.0) THEN
            x1=(-b+sqrt(d))/(2.0*a)
            x2=(-b-sqrt(d))/(2.0*a)
            PRINT *,'两个不等实根：'
            PRINT *,'x1=',x1
            PRINT *,'x2=',x2
        ELSE IF(d==0.0) THEN
            PRINT *,'两个相等实根'
            PRINT *,-b/(2.0*a)
        ELSE
            pr=-b/(2.0*a)
            pi=sqrt(-d)/(2.0*a)
            PRINT *,'两个复根'
            PRINT *,'x1=',pr,'+',pi,'i'
            PRINT *,'x2=',pr,'-',pi,'i'
        END IF
    END IF
    END
```

案例 2-已知 U，V 风速，判断风向

```
    program ex0308
    real u,v
    read *, u,v
    if(u>0.0) then
        if (v>0.0) then !u>0,v>0
            print *, '西南风'
        else if (v<0.0) then ! u>0, v<0
            print *, '西北风'
        else ! u>0,v=0
            print *, '西风'
        end if
    else if (u<0.0) then
        if (v>0.0) then !u<0,v>0
            print *, '东南风'
        else if (v<0.0) then ! u<0, v<0
            print *, '东北风'
        else ! u<0,v=0
            print *, '东风'
        end if
    else
        if (v>0.0) then !u=0,v>0
            print *, '南风'
        else if (v<0.0) then ! u=0, v<0
            print *, '北风'
        else ! u=0,v=0
            print *, '无风'
        end if
    end if
    end
```

3.3 循环结构程序设计

3.3.1 循环语句的形式

3.3.1.1 有循环变量的 DO 循环结构

一般形式　**[结构名]　DO　循环变量 = E1，E2，E3**
　　循环体
　　END DO [结构名]

变量分别为：起始值、终止值、步长
不一定取到终止值，例如步长 2，终止值 9

案例

```
    DO   I=1,3,2              起始值 1，终止值 3，步长 2
        M=I*I                 这段循环会执行两次
        PRINT*, I, M
    END   DO
```

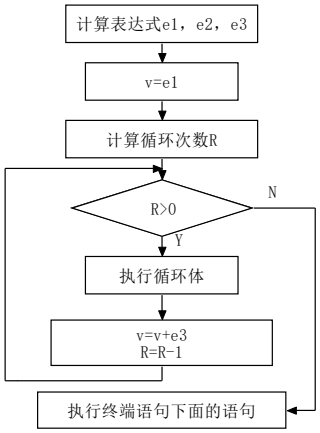
DO 循环结构循环的次数

① REAL :: I	DO	I=0.6, 6.6, 1.4	执行 5 次，0.6, 2.0, 3.4, 4.8, 6.2
② INTEGER:: I	DO	I=0.6, 6.6, 1.4	执行 7 次，0, 1, 2, 3, 4, 5, 6

③ DO　R=0.6, 6.6, -1.4　　执行 0 次
④ DO　R=6.6, 0.6, 1.4　　执行 0 次

循环次数　**R = MAX (INT ((E2 - E1 + E3) / E3), 0)**

执行过程
① 先计算 E1、E2、E3 的值，然后转换为与循环控制变量相同的类型
② 给循环控制变量赋初值 v= E1
③ 首先**计算循环次数**
④ 检查循环次数，当 R>0 时，执行循环体，继续做第 4~6 步；当 R≤0 时，转向第七步，直接循环结束。
⑤ 当执行到循环终端语句时，循环变量按步长增值，即：循环变量 +E3
⑥ 循环次数减 1：R = R – 1 ；返回 第 4 步继续执行。
⑦ 循环结束



注意事项
① 循环变量在循环体中可以引用，但不允许重新赋值。
② E1、E2、E3 的类型应与循环变量相同。
③ E3 的缺省意味着循环步长为 1。
④ 可以不经过 END DO 语句退出循环，例如 STOP 可以直接退出循环。
⑤ DO 循环和其它结构(如块 IF 结构、CASE 结构)可以相互嵌套使用。
⑥ 退出循环后，循环变量仍然存在。

案例

循环输入一周日最高气温，判断最高气温，并计算一周平均最高气温。 数据为 2014 年 3 月 23 日-29 日南京日最高气温如下：16.0 17.0 17.0 18.0 16.0 22.0 24.0 （单位：℃）。

```
real t,tmax,sum,tave;          Integer I
tmax=0.0 ;                     sum=0.0
do i=1,7
    read *, t
    sum=t+sum
    if (t>=tmax) then
        tmax=t
    end if
end do
tave=sum/7.0
print *, 'tmax=', tmax;        print *, 'tave=', tave;          end
```

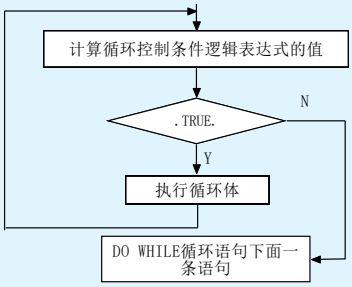
3.3.1.2 DO-WHILE 控制的循环结构

一般形式　**[结构名]　DO WHILE （逻辑表达式）**
循环体　　必须存在一句控制循环变量重新变化的语句
END DO [结构名]

案例

循环输入每六小时降水资料，如果发现数据小于 0 或者大于 1000 时，终止循环，并提示输入数据异常。

```
real precip            print *, '输入每六小时降水量(mm):'
read *, precip
 do while(precip.ge.0.and.precip.le.1000)
     print *, '降水量(mm)：', precip
     read *, precip        此处必须要重新获取数据
end do
print *, '输入数据异常'
end
```



结构选择　① 如果已知循环的次数，或者知道循环的初值和终值，那么用有循环变量的 DO 循环结构。

② 知道循环的条件时，选用 DO WHILE 循环结构，或者是重复 DO 循环结构。其中在使用这两种结构时，注意循环条件的变化，以避免“死循环”。

3.3.2 循环控制语句

3.3.2.1 EXIT 语句

用法　EXIT 语句用于强制退出循环，将执行控制转移到当前循环或结构之外。

一般格式　**EXIT**　[DO 循环结构名]

使用说明　EXIT 语句通常是作为逻辑 IF 语句的内嵌语句来使用，其作用是有条件中断。

形式为：**IF(逻辑表达式)　EXIT [结构名]**

执行过程　当逻辑表达式为真时，中止正在执行的循环，将控制转到 EXIT 语句指定的结构之后；当逻辑表达式为假时，继续正在执行的循环，不进行任何转移。

案例

输入正整数n，求级数的前n项和，如果当某项绝对值$\leq 10^{-5}$时，虽未满n项，也因满足精度而不再加入下一项。

```
INTEGER :: i,n
REAL :: s=0,t
READ *, n
DO i=1,n
    t=1./(i*(i+1))                  注意这里的 1. 必须为实型
    s=s+t
    IF(ABS(t)<=1.E-5)  EXIT         循环的非正常出口时，I 的值为当前值
END DO
IF (i==n+1) i=i-1                   循环正常出口时，I 的值为终值+步长
PRINT *, 'SUM=',s,' TERM=',i
END
```

3.3.2.2 CYCLE 语句

用法　CYCLE 语句用于中止执行循环体中剩余的语句，重新执行下一轮循环。相当于 `continue`。

一般格式　**CYCLE**　[DO 循环结构名]

使用说明　① CYCLE 语句与 EXIT 语句不同，它不中止循环的执行，而是将循环变量增加一个步长，从下一个循环开始执行。

② CYCLE 语句通常是作为逻辑 IF 语句的内嵌语句来使用。

形式为：**IF(逻辑表达式)　CYCLE　[DO 循环结构名]**

执行过程　当逻辑表达式为真时，中止正在执行的循环体的剩余语句，将控制转到循环体的开始重新循环；当逻辑表达式为假时，继续正在执行的循环，不进行任何转移。

案例 1

顺序输出 1-10 序列中除了 9 以外的其它数字

```
do i=1,10
    if (i==9)  cycle     跳出第 9 个
    print *, i
end do
```

案例 2

输入 3 月份气温，缺测记录为 999，请统计 3 月份平均气温。

```
real t,tsum,tave
integer i,num
tsum=0.0
```

```
num=0
do i=1,31
    read *,t                    另一种思路
    if(t==999.0) cycle          if(t/=999.0) then
    tsum=tsum+t                     tsum=tsum+t
    num=num+1                       num=num+1
end do                          end if
tave=tsum/num
print *, tave
```

3.3.3 循环语句的嵌套

一般格式 即在正常的循环中嵌套一个循环。

循环次数 外层循环的次数为：R1　　内层循环的次数为：R2

整个循环的次数为：**R=R1*R2**

说明

① 三种 DO 循环结构必须是完整的嵌套。

② 循环嵌套时，内、外层不能使用相同的循环变量。

③ 循环转移的问题，比如 EXIT 和 CYCLE。

④ DO 循环结构可以与选择结构的嵌套。

循环次数的计算

```
M=0                         M 记录循环次数
DO I=1,3,4                  外层循环 1 次
    DO J=4,19,4             内存循环 4 次
        M=M+1               总循环 4 次
        PRINT *, M
ENDDO

M=0
II: DO I=1,3,4              外层理论循环 1 次
    JJ: DO J=4,19,4         内层理论循环 4 次
        M=M+1
        if(M>=3)EXIT II     当 M 大于等于 3 时，结束外层循环。如果换成 CYCLE，此处不变。
        END DO   JJ         因此总循环次数为 3 次
ENDDO II
PRINT*,M
```

3.3.4 两种循环形式的比较和关系

主要关系 ① 带循环变量的 DO 循环用来处理已确定循环次数的问题。DO WHILE 循环既可以用来处理已知循环次数的循环问题，也可用来处理不确定循环次数的问题。

② 对事先已确定循环次数的问题，用带循环变量的 DO 循环比较方便，它能使循环变量自动增值，不需用户写逻辑表达式，只需写出循环变量的初值、终值和步长即可，使用方便。因此带变量的 DO 循环在气象上使用最为频繁。

③ 由于带循环变量的 DO 循环只能判断处理一个条件（循环次数 R），当需要多个入口条件时，应考虑 DO WHILE 循环结构。

④ 各类循环可以相互转换以及互相嵌套。嵌套时，需要完整嵌套。

应用实例

假设降水资料的存放路径按照右图有规律地存放，请利用循环，将 1979 年 1 月-1990 年 12 月的资料路径，依次输出至屏幕上。

```
integer iy,im                    定义两个循环变量
character year*4, mo*2           定义两个待赋值的字符型变量
character dir*100                存放整个路径的长为 100 的变量
dir 变量由于其有效长度是变化的，故定义一个足够长的字符串
do iy=1979,1990
    write(year(1:4),'(i4)') iy       把整型数据写到字符型变量之中
    mo='  '
    此处为情况保留值，将 mo 赋值为 2 个或者 1 个空格均可以
    do im=1,12                       数值型数据无法直接放到字符串里
        if(im<10) then
            write(mo(1:1),'(i1)') im
        else
            write(mo(1:2),'(i2)') im
        endif
        dir='e:\data\'//trim(year)//'\'//trim(mo)//'\precip'
        print *, trim(dir)
    enddo
enddo
```

```
1 e:\data\1979\1\precip
2 e:\data\1979\2\precip
3 e:\data\1979\3\precip
4 e:\data\1979\4\precip
5 e:\data\1979\5\precip
6 e:\data\1979\6\precip
7 e:\data\1979\7\precip
8 e:\data\1979\8\precip
9 e:\data\1979\9\precip
10 e:\data\1979\10\precip
11 e:\data\1979\11\precip
12 e:\data\1979\12\precip
13 e:\data\1980\1\precip
14 e:\data\1980\2\precip
15 e:\data\1980\3\precip
16 e:\data\1980\4\precip
17 e:\data\1980\5\precip
18 e:\data\1980\6\precip
19 e:\data\1980\7\precip
20 e:\data\1980\8\precip
21 e:\data\1980\9\precip
22 e:\data\1980\10\precip
23 e:\data\1980\11\precip
24 e:\data\1980\12\precip
25 ......
```

如果不使用 if 块处理，只用 write(mo(1:2),'(i2)') im 语句，所得运行结果：

```
e:\data\1989\ 7\precip
e:\data\1989\ 8\precip
e:\data\1989\ 9\precip
e:\data\1989\10\precip
e:\data\1989\11\precip
e:\data\1989\12\precip
e:\data\1990\ 1\precip
e:\data\1990\ 2\precip
e:\data\1990\ 3\precip
e:\data\1990\ 4\precip
e:\data\1990\ 5\precip
e:\data\1990\ 6\precip
e:\data\1990\ 7\precip
e:\data\1990\ 8\precip
e:\data\1990\ 9\precip
e:\data\1990\10\precip
e:\data\1990\11\precip
```

注意

循环的条件往往是程序易错之处，需要合理估算你的程序运行时间，例如,如果设计的循环程序要求计算量较大，但是运行过快，需要检查循环是否执行，多重嵌套时需要确保每层循环都按照设计思路进行运行。