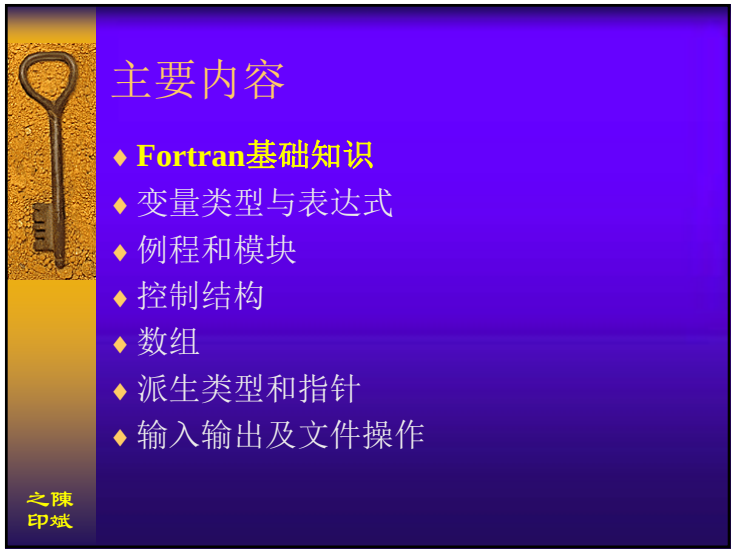
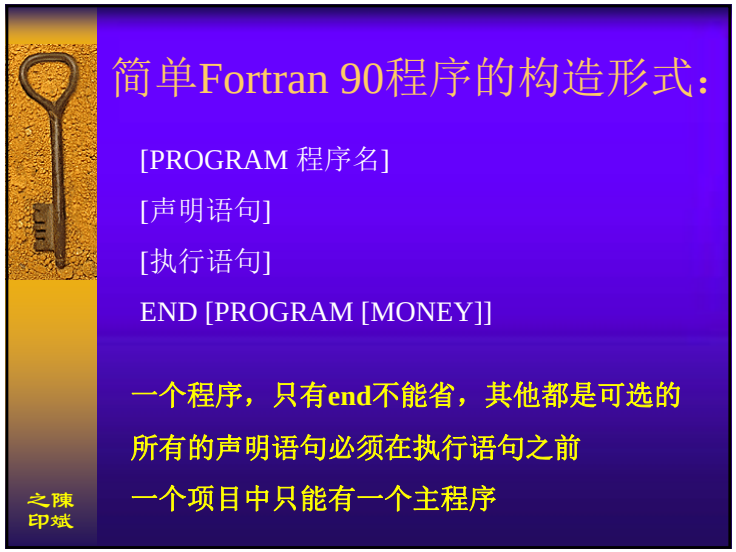
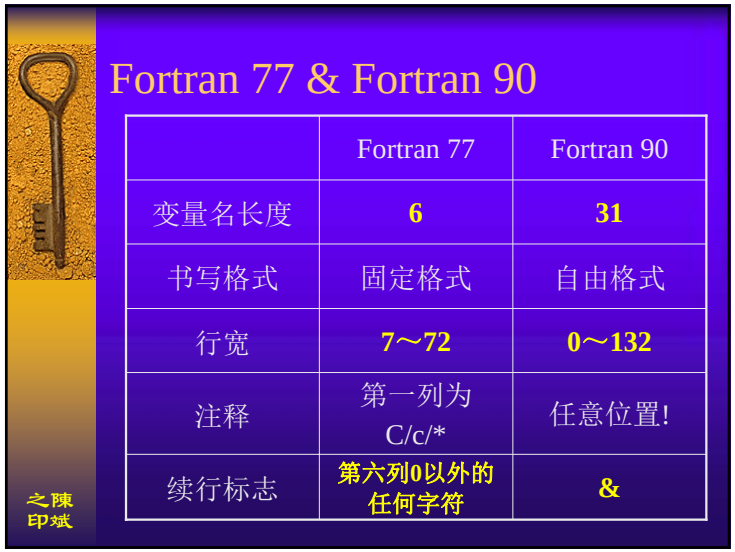






	Fortran 77	Fortran 90
变量名长度	**6**	**31**
书写格式	固定格式	自由格式
行宽	**7～72**	**0～132**
注释	第一列为C/c/*	任意位置!
续行标志	**第六列0以外的任何字符**	**&**

字符集（character sets）

- 允许出现在一个语言的程序里的字符的全体
- 26个大写字母：A ~ Z
- 26个小写字母：a ~ z
- 10个数字： 0 ~ 9
- 下划线： _
- 5个运算符号： + - * / （**）
- 特殊字符： () = . , ; : ’ ” $ ！ % & < > ? &
- 空格 “ ”

之陳印斌

标识符命名

- 给变量、常量、函数等标识符命名时，必须以字母开头，后面可接多达30个字母（A~Z、a~z）、数字（0～9）或下划线（_）
  - **只能以字母开头；**
  - **不能含有空格；**
  - **不区分字母大小写；**
  - **长度限定为31个字符**（Fortran 77为6个字符）；
  - 避免与关键字、标准例程重名。

之陳印斌

变量名（Variable name）

- 有效的变量名

area
distance
z123456789
long_axis_of_elliptic_circle
a3b4c5
abstract_3
TimeAndSpace
century____

- area，AREA，aREa等等都是一个变量

之陳印斌

变量名（Variable name）

- 无效的变量名

This_is _a_very_long_variable_name
3_day
$600
my-help
exchange rate
U.S.A
Ask?
“UK”
b/a
engry&power
wang@163.com
fluent 6.3

之陳印斌

语句（Statement）

- 语句是Fortran程序的基本单位，一条语句可包含0～132个字符；
- 除赋值语句外，所有的语句都从一个关键字开始；
- 一般情况下，每行一条语句；
- **如果一行有多条语句，它们之间以分号分隔**
- 假如一条语句一行写不完，允许出现续行，但要求被续行最后的非空白字符为“**&**”
- 续行从下一行（非注释行）的第一个非空白字符开始，如果下一行的非空白字符为“&”，则续行从该字符后的第一个字符开始。
- **Fortran 90 允许出现多达39个续行**。

之陳印斌

Example

- 正确的写法
- axis1=1; axis2=2; axis3=3
- Area = sqrt(half * (half - axis1) * (half - **&** axis2) * (half - axis3))
- Area = sqrt(half * (half - axis1) * (half- **&** **&** axis2) * (half - axis3))

- 错误的写法
- axis1=1, axis2=2, axis3=3
- Area = sqrt(half * (half - axis1) * (half - **&**axis2) * (half - axis3))

之陳印斌

引号的使用

a=“Hello”！FOTRAN 90可以用双引号界定字符串

b=‘Hello’！FOTRAN 77只能用单引号界定字符串

c=“That’s right.”！**用双引号界定字符串，可以在字串中任意使用单引号**

d=‘That’’s right’ !**用单引号界定字符串时，输出单引号要连续使用两个单引号**

e=“That’s ““right””.” !**用双引号界定字符串时，输出双引号也要连续用两个双引号**

之陳印斌

声明

- Example：

implicit integer(A,B,C) ！A,B,C开头的变量都视为整数

implicit integer(A-F,I,K) ！A到F及I,K开头的变量都视为整数

implicit real(M-P) ！M到P开头的变量都视为浮点数

implicit none ！关闭隐含规则，所有的变量都要事先声明

- 注意：
  - **IMPLICIT指令要马上接在PROGRAM指令的下一行**，不能放在其它位置

之陳印斌

主要内容

- Fortran基础知识
- **变量类型与表达式**
- 例程和模块
- 控制结构
- 数组
- 派生类型和指针
- 输入输出及文件操作

之陳印斌

数据类型

- 固有数据类型
  - 数值型 (numerical)
    - 整型　INTEGER
    - 实型　REAL
    - 复数型 COMPLEX
  - 非数值型
    - 字符型 CHARACTER
    - 逻辑型（布尔型） LOGICAL
- 自定义数据类型（派生数据类型）

之陳印斌

变量声明及其初始化

- **声明部分必须出现在执行部分之前，而不能将声明语句插在执行部分之中。**
- Fortran 90：
- 数据类型 **[[, 属性]::] 变量列表**
  - DIMENSION、PARAMETER、TARGET、
  - POINTER、ALLOCATABLE、INTENT
  - **::可以省略，但如果在声明的同时给变量赋初值，则不能省略**

之陳印斌

找错误

- real(8) parameter a
- integer(4) parameter ::b
- Complex, parameter c=(1.0,2.0)

- real(8), parameter a
- integer(4), parameter ::b
- Complex, parameter **::** c=(1.0,2.0)

之陳印斌

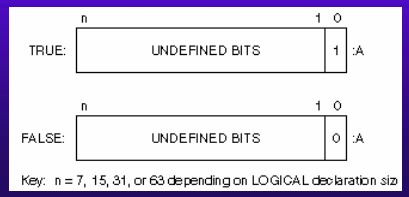
复数型（COMPLEX）

- 复数型变量
  - (r, i)
  - 复数型常量的类型参数，**取实部和虚部的实数种类参数的极大值**
- 复数由实部和虚部组成，z=x+yi，x为实部，y为虚部，x、y皆为实型变量
- 设定
- **z = (x, y)　　! a = x + iy**
- **z = (1.5, 2.5)　　! a = 1.5 + 2.5i**
- **z = 1.5　　! a = 1.5 + 0i**
- **Z= (0, 2.5)　　! a= 0 + 2.5i**

之陳印斌

逻辑型（LOGICAL）

- 逻辑型变量
  - LOGICAL L
  - LOGICAL([KIND=]N) L
- 逻辑型常量
  - .TRUE.（逻辑真）
  - .FALSE.（逻辑假）



之陳印斌

字符型（CHARACTER）

- 字符型变量
  CHARACTER [([LEN=]len)]
  CHARACTER [([LEN=]len [, [KIND=]n])]
  CHARACTER [(KIND=n [, LEN=len])]
- 字符串有两个可选参数：长度参数、种类参数
- 种类参数总是1，即一个字符占一个存储字节
- 假如两个可选参数都没给出，缺省值取1： character c
- 若只给出一个参数，则代表长度参数：character(20) c
- 若给出两个参数，则依次为长度参数和种类参数（种类参数智能为1）： character(20, 1) c
- 若采取关键字声明形式，则参数的顺序可以任意：
  character(KIND=1, LEN=20) c

之陳印斌

例子

- CHARACTER (10) ::STR=“HIGH SCORE”
- CHARACTER ::K=“T”
- CHARACTER(4) C
- C=“XJTU”
- CHARACTER ::C
- C=“w”

之陳印斌

字符型（CHARACTER）

- 字符型常量
  - 统一以单引号（‘’）或双引号（“”）为界定符
  - [k_]’ch’
  - [k_]”ch”
  - Ch：Fortran字符集之内或之外的字符，字符的个数为字符串的长度
  - ‘Today ‘’s date is 2007-05-14’
  - **用连续的两个界定符表示与其相同的字符**
  - 界定符必须统一，要么都是单引号，要么都是双引号
  - 字符串中字符的最大数量：32767

之陳印斌

例子

- “WHAT’S YOUR NAME?”
- ‘WHAT’’S YOUR NAME?’
- “ ”
- “Can I help you?”
- “New world record!!!”

之陳印斌

算术表达式

- 运算符
- \+ -
- * /
- **
- （）

优先级增加

- 运算级相同：从左至右
- 连续的乘幂运算：从右至左
- **计算的结果是把“=”右边的结果传给左边的变量**
- **整数之间的除法，结果也是整数！**
- **如果不能整除，小数部分无条件舍去！**

之陳印斌

算术表达式

23**2**　　2A：**2*A**

arcsinx：**ASIN(X)**

(A+B)(C+D)：**(A+B)*(C+D)**

$\frac{(A+B)(C+D)}{2(E+F)}$ ：**((A+B)*(C+D))/(2*(E+F))**

$d=\sqrt{x^2+y^2+z^2}$　　**d=sqrt(x**2+y**2+z**2)**

之陳印斌

Example

```
program convert
        real a                  !Integer a
        a = 10 / 4 / 0.5
        print *, 'a = ', a
end
```

之陳印斌

关系运算符

F90	F77	功能/意义
==	.EQ.	判断是否「等于」
/=	.NE.	判断是否「不相等」
>	.GT.	判断是否「大于」
>=	.GE.	判断是否「大于或等于」
<	.LT.	判断是否「小于」
<=	.LE.	判断是否「小于或等于」

之陳印斌

例子

- A=3　　!给A赋值为3
- A==3　　！判断A是否等于3
- if(A=3) n=n+1　　!这句话的意思是如果把A赋值为3是真，则n加1，逻辑上讲不通，所以编译的时候会报错，正确的写法应该是
- if(A==3) n=n+1　　！如果A等于3，则n加1

之陳印斌

逻辑运算符——两个逻辑运算式间的运算关系

.AND.	交集，如果两边的式子都成立，整个条件就成立
.OR.	或集，两边的式子只要有一个成立，整个条件就成立
.NOT.	逻辑非，如果后面的式子不成立，整个式子就算成立
.EQV.	两个式子的逻辑运算结果相同时，整个式子就成立
.NEQV.	两个式子的逻辑运算结果不同时，整个式子就成立

之陳印斌

关系表达式

- 关系表达式的结果是一个逻辑量
- **.TRUE.** 或者**.FALSE.**,输出时为**T**或者**F**
- **Print*，3.GT.5　!输出为F**
- 关系表达式允许不同类型的算术表达式比较
- **Print*, 8.EQ.(5/2+1.5)*2** !输出结果为**F**
- 比较两个实数大小时，由于机内表示为近似值，所以不应当做两个数来比较
- **ABS(A-B).LE.1E-06**
- !表示A和B误差的绝对值小于10^{-6}时，认为二者相等

之陳印斌

例子

- 0.0<x<5.0　或 x>=10
- **X.GT.0.0.AND.X.LT.5.0 .OR. X>=10.**
- A+B≠0且C=A-B
- **A+B/=0 .OR. C==(A-B)**
- 计算下面逻辑变量的值
- A=16.3; B=12.3; C=10; L=.TRUE.
- A+B.GT.C.AND.B+C.LT.A.OR.B.GT.C
- **T.AND.F.OR.T → F.OR.T → T**

之陳印斌

例子

- 写出10以内的素数的逻辑表达式
- 分析：
  - 要求的数应该同时满足以下条件，即为并的关系
    - 大于1
    - 小于10
    - 除以2余数不为0
    - 除以3余数不为0
- **N.GT.1.AND.N.LT.10.AND.MOD(N,2).NE.0.AND.MOD(N,3).NE.0**

之陳印斌

例子

- 写出1000以内的含有3或者能被3整除的整数
- 分析：
  - 要求的数应该同时满足，即为并的关系，应该用.AND.
    - 大于0
    - 小于1000
  - 并且至少满足以下条件之一，即为或的关系，应该用.OR.
    - 能被3整除
    - 个位含有3
    - 十位含有3
    - 百位含有3
- **INTEGER N**
- **N.GT.1.AND.N.LT.1000.AND.MOD(N,3).EQ.0.OR.MOD(N,10).EQ.3.OR.N/10.EQ.3.OR.N/100.EQ.3**

之陳印斌

主要内容

- Fortran基础知识
- 变量类型与表达式
- **例程和模块**
- 控制结构
- 数组
- 派生类型和指针
- 输入输出及文件操作

之陳印斌

主程序

- 一个完整的程序有且只有一个主程序
- **注意顺序**

```
[PROGRAM   程序名]
        [声明语句]
        [执行语句]
        [USE MODULE 模块名]
        [CALL 外部例程]
[CONTAINS
        内部例程]
END [PROGRAM[程序名]]
```

之陳印斌

主程序

- 主程序使用时应注意以下几点：
  - **主程序只有END语句是必须的，其它都是可选的；**
  - 若含有内部例程，则必须有CONTAINS关键字；
  - 可以有多个内部例程，但不允许内部例程的嵌套；
  - **主程序的END语句若出现程序名，其前面的PROGRAM关键字不能少。**

之陳印斌

改错

- Program abc(x,y)
- End program abc

Program abc
End program abc

- Program
- End program

Program xy
End program

- Program xyz
- End xyz

Program xyz
End
(END Program xyz)

之陳印斌

内部函数—构造形式

- 内部函数位于主程序的CONTAINS关键字和END语句之间，构造形式为：

  FUNCTION 函数名（[参数列表]）
  [声明语句]
  [执行语句]
  END FUNCTION [函数名]

- 强制类型声明IMPLICIT NONE，作用域为整个程序单元，无须再重写。

之陳印斌

判断正误

- Program
  Contains
  function f(x)
  ! function
  ! function f
  real f,x
  end function
  ! end function f
  ! end function f(x)
  ! End
  End program

应该这样写 function f(x)
在没有参数的情况下也可以写成 function f()

这两个是对的
end function
end function f

之陳印斌

语句函数

- 定义语句函数的形式为：
  - 函数名（参数1，参数2，…）=函数表达式
- 语句函数在使用时应注意：
  - **语句函数先定义后使用，且只能用一条语句来定义；**
  - **定义语句应放在声明部分，且放在语句函数相关的类型声明之后**
  - 参数列表可以为空（此时，函数实际为一常量表达式），但函数名后面的一对括号，无论在定义还是引用时都不能省略

之陳印斌

例子

- PROGRAM SQUART
- REAL F,X,Y
- F(X,Y)=SQRT(X**2+Y **2)
- REAL RESULT,A,B
- A=3.0;B=5.0
- RESULT=F(A,B)
- PRINT*, RESULT
- END

程序中F(X,Y)声明了一个语句函数，是声明语句，不是执行语句，所以在其之后还可以声明变量

之陳印斌

内部子程序

- 子程序和函数的主要差别:
  - 没有返回值和子程序名关联，因此无需声明子程序类型;
  - **通过CALL语句调用子程序;**
  - 在例程原型（头）和END语句（尾）中，使用关键字SUBROUTINE;
  - **若子程序参数表为空，子程序名后的一对括号可以省略。**
- **通常函数通过函数名返回一个值，而子程序通过参数可以返回多个值。**

之陳印斌

例子

```
program fly3
implicit none
real v0,a,t,s
real,parameter :: g=9.8,pi=3.14
print*,'此程序是使用内部子程序计算斜抛运动的飞行时间'
print*,' 请输入初速度v0和仰角θ '
read*,v0 ,a
call d(a,v0,t,s)
! call d(a,v0)
 PRINT*,'飞行时间为',t
 print*,'飞行距离为',s
 contains
 subroutine d(a,v0,t,s)
! subroutine d(a,v0)
 real s,v0,t,a
 !real v0, a
 t=2*v0*sin(a*pi/180)/g
 s=v0*cos(a*pi/180)*t
  PRINT*,'飞行时间为',t
 print*,'飞行距离为',s
 end subroutine
end program fly3
```

!把t,s放在变量列表里，其值就可以在主程序和子程序中间传递,主程序可以输出结果

!对内部子程序，主程序中的变量可以直接调用，不需要再声明，如果声明则会认为是新的变量
!外部子程序中用到的变量都要重新声明

之陳印斌

外部例程

- 通用的例程一般作为外部例程来实现，以便被多个调用程序调用。
- 通常，外部例程位于单独的文件中，除了头、尾部分外，外部例程和主程序在形式上是相同的。
- 外部例程和内部例程的主要差别：
  - **外部例程可以含有内部例程，而内部例程不能再含有内部例程;**
  - **END语句中的关键字FUNCTION/SUBROUTINE，在外部例程中是可选的，但在内部例程中是必须的。**

之陳印斌

参数传递

```
SUBROUTINE  SUB(X,Y,Z)
      REAL,INTENT(IN)          ::X
      REAL,INTENT(OUT)       ::Y
      REAL,INTENT(INOUT)   ::Z
      ……
END SUBROUTINE  SUB
```

其中 INTENT(IN) 表示向例程传入数据，拥有该属性的虚参不允许改变

INTENT(OUT) 表示传出数据，对应的实参必须是一变量

INTENT(INOUT) 表示传进/传出数据，对应的实参也必须是一变量

之陳印斌

例子

```
PROGRAM MAIN
    IMPLICIT NONE
    REAL::X
    X=3.5
    CALL SUB(4.0,X,X)
    PRINT*, X
    END

SUBROUTINE  SUB(X,Y,Z)
    IMPLICIT NONE
    REAL X, Y, Z
    Y=Y+X
    Z=Y+Z
END SUBROUTINE SUB
```

程序运行结果是：（ ）

A) 12.000000　B) 15.00000　C) 10.500000　D) 11.000000

之陳印斌

全局变量与局部变量

- **在内部例程中声明所有变量**应成为一条编程规则，以消除全局变量带来的负面影响。
- **如果需要从程序单元向内部例程传递数据，最安全的方式是利用参数传递。**
- 如果需要在多个内部例程中共享大量数据，最好的解决方案是在模块中统一声明全局变量，需要访问这些全局变量的内部例程引用该模块即可。

之陳印斌

主要内容

- Fortran基础知识
- 变量类型与表达式
- 例程和模块
- **控制结构**
- 数组
- 派生类型和指针
- 输入输出及文件操作

之陳印斌

IF语句和块IF结构混淆使用：

IF语句构造形式为：

- IF(condition) statement
- **IF语句只能写在一行上**

- IF块的构造形式：
  - IF(logical-expr1) THEN
  - block1
  - ELSE IF(logical-expr2) THEN
  - block2
  - ELSE IF(logical-expr3) THEN
  - …
  - blockE
  - END IF

之陳印斌

选择结构中几种典型的错误(1)

```
if(x>0) then
   print*,‘正数’
else if(x<0 ) then
   print*,‘负数’
end if
```

之陳印斌

选择结构中几种典型的错误(2)

```
if(x>0)  then
print*,‘正数’
else if(x<0 )  then
print*,‘负数’
 else
print*,‘zero’
end
```

之陳印斌

选择结构中几种典型的错误(3)

```
if(x>0) print*,‘正数’
else if(x<0 )print*,‘负数’
end
```

之陳印斌

确定性DO循环

- 确定性DO循环的构造形式：
  - [name:]DO 变量=表达式1，表达式2[,表达式3]
  - 语句块
  - END DO[name]
- 循环次数的确定：

$$MAX\left(\frac{\text{表达式2}-\text{表达式1}+\text{表达式3}}{\text{表达式3}},0\right)$$

- **系统在进行确定性循环时，先按上述公式计算循环次数。如果循环变量的步长为0，就会发生除0的错误。**

之陳印斌

- 假如将DO循环简记为：
  - DO I=A, B, C
- 根据具体情况，DO循环可分为以下4种：
  - 1）C>0, A<B。执行循环体，循环变量从A开始，第二次循环以后，每次增加一个步长C，直到循环变量的终值超过B。
  - 2）C>0, A>B。不执行循环体。
  - 3）C<0, A>B。执行循环体，循环变量从A开始，第二次循环以后，每次减小一个步长的绝对值，直到循环变量的终值小于B。
  - 4）C<0, A<B。不执行循环体。
- **如果在循环结构执行后要引用循环变量，请注意此时循环变量的值：第（1）种情况，I>B；第（3）种情况，I<B。**

之陳印斌

例子

- DO N=1,10,4　!程序运行时，N的值依次为1，5，9，到N=13时跳出循环
- DO I=10,-10,-2　！到n=-12时跳出
- DO I=3,8,-2　!因为不满足条件，所以程序不会执行，最终I的值仍然为3

之陳印斌

Example4.2：　4.2.F90

```
DO I=2,7,2
    WRITE(*,'(I3)',ADVANCE='NO')I
END DO  !输出：2 4 6
I = ?

DO I=5,4
    WRITE(*,'(I3)',ADVANCE='NO')I
END DO  !不执行循环
I = ?

DO I=5,1,-1
    WRITE(*,'(I3)',ADVANCE='NO')I
END DO  !输出：5 4 3 2 1
I = ?

DO I=1,6,-2
    WRITE(*,'(I3)',ADVANCE='NO')I
END DO  !不执行循环
I = ?
```

之陳印斌

例子

```
DO I =1, 10
      PRINT*, I, Fact(I)        ！主程序循环次数由前面公
END DO                          式计算为10
 CONTAINS  FUNCTION Fact(N)
          INTEGER Fact, N, Temp
            Temp = 1
              DO I = 2, N
                Temp = I * Temp
              END DO
              Fact = Temp
          END FUNCTION Fact
```

I是全局变量，当函数Fact第一次被引用时，I=1,该值被传递给虚参N，相同的I在函数循环体中被赋值2，此时2>N，不执行循环，当Fact返回主程序打印时，I的值为2，下次引用时，I在主程序DO循环中变为3，依次类推，程序不会计算偶数阶乘

之陳印斌

几种非确定性DO循环

- 第一种：
```
DO
    IF (logical-expression) EXIT
        block
END DO
```
- 第二种：
```
DO
    block
    IF(logical-expression )EXIT
END DO
```
- 第三种：
```
DO  WHILE(logical-expression)
    block
END DO
```
- Fortran90中的EXIT命令，用于跳出DO循环。原则上，EXIT可以放在DO循环的任何位置，但为使代码更具可读性，通常将EXIT放在DO循环头部，或放在DO循环末尾。

之陳印斌

例子

- Do
- if(n==0) exit
- n=n-1
- End do

- Do while (n.eq.0)
- n=n-1
- End do

之陳印斌

主要内容

- Fortran基础知识
- 变量类型与表达式
- 例程和模块
- 控制结构
- **数组**
- 派生类型和指针
- 输入输出及文件操作

之陳印斌

数组声明

- Fortran90数组声明的一般形式为
  - **TYPE,DIMENSION([dl:]du[,[dl:]du]…) ::Arr**
  - **TYPE [::]Arr([dl:]du[,[dl:]du]…)**
- TYPE代表数据类型，dl和du分别为维的上界和下界，Arr为数组名（数组变量）
- 数组声明实例：
  - REAL,DIMENSION(15) ::X ！下界缺省为1
  - REAL,DIMENSION(1:5,1:3) ::Y
  - REAL,DIMENSION(-4:0,1:3) ::Z
- 维数、上下界、大小、形状

之陳印斌

数组的存储规则

- 所有元素都是存储在内存的同一个连续区块当中，**多维数组为列主存储column major**

real A(5),B(0:4),C(3,3)

- **A(1)→A(2)→A(3)→A(4)→A(5)**
- **B(0)→B(1)→B(2)→B(3)→B(4)**
- **C(1,1)→C(2,1)→C(3,1)→C(1,2)→C(2,2)→C(3,2)→C(1,3)→C(2,3)→C(3,3)**

之陳印斌

数组赋初值

- 使用DATA的一般形式为：
  - INTEGER A(5)
  - DATA A /1,2,3,4,5/
  - !A(1),A(2),A(3),A(4),A(5)的值分别为1，2，3，4，5
- **DATA的数据区中，还可以使用乘号“*”来表示数据的重复（重复次数写在前面）**
  - DATA A /5*3/
  - ! A(1), A(2), A(3), A(4), A(5) 的值均为3

之陳印斌

数组的存储规则（两个错误）

```
Real c(3,3)
Integer I, j
Data c (1,2,3,4,5,6,7,8,9)
  Do i=1,3
     Do j=1,3
        Print (*,*) c(i, j)
     End do
  End do
end
```

之陳印斌

使用隐式循环给数组赋初值

- 隐式循环可以用来设置数组的初值
  - DATA (A(I),I=2,4) /2,3,4/
  - !A(2),A(3),A(4)的值分别为2，3，4
- **隐式循环可以认为是DO循环的简略形式。在隐式循环中同样可设置循环变量的增量(步长)**
  - DATA (A(I),I=1,5,2) /1,3,5/
  - !A(1),A(3),A(5)的值分别为1，3，5
- **隐式循环也可以嵌套**
  - INTEGER B(2,2),I,J
  - **DATA ((B(I, J), I=1, 2), J=1, 2)) /1,2,3,4/**
  - **PRINT*,((A(M, N), M=1, 3), N=1, 3)**

之陳印斌

数组整体操作

- Fortran90中可以对数组进行整体操作，大大简化了其它语言需要使用循环才能完成的操作。下面是几个例子
  - A=5
- 其中，A是任意维数及大小的数组。该语句将数组A所有元素的值设为5。
  - **A=(/1,2,3/)**
- 其中，A(1)=1,A(2)=2,A(3)=3。所提供的数据个数必须跟数组A的大小一样。
  - A=B
- 其中，A和B是形状完全相同的数组。该语句将数组A相应位置的元素的值设置成同数组B。

之陳印斌

- 其它的几个例子：
  - A=B+C
  - A=B-C
  - **A=B*C ！非矩阵相乘**
  - A=B/C
- 其中，A,B,C是三个完全相同的数组。**上述语句分别将数组B和C相应位置的值相加、相减、相乘和相除，得到的结果赋给数组A对应元素。**
  - A=sin(B)
- 将数组A的每一个元素赋为数组B相应元素的sin值，数组B须是实数型。

之陳印斌

例子

  - A=B>C
- A、B和C是三个形状完全相同的数组。不过，A为逻辑型数组，B和C为同类型的数值型数组。对一维数组，其等价于：
  - DO I=1,N
  - IF(B(I)>C(I)) THEN
  - A(I)=.TRUE.
  - ELSE
  - A(I)=.FALSE.
  - END IF
  - END DO

之陳印斌

数组段操作

- 数组段的下标三元组形式为：
  - [<bound1>]:[<bound2>][:<stride>]
- 数组段起始于下标bound1，终止于bound2，步长为stride。
  - A(:) ！整个数组
  - A(3:9) ！A(3)-A(9), 步长为1
  - A(9:3:-1) ！A(3)-A(9), 步长为-1
  - A(m:n) ！A(m)-A(n),步长为1
  - A(m:n:k) ！A(m)-A(n), 步长为k
  - A(m:) ！A(m)-A(上界), 步长为1
  - A(:n) ！A(下界)-A(n), 步长为1
  - A(::2) ！A(下界)-A(上界), 步长为2
  - A(m:m) ！1个元素的数组段

之陳印斌

- 数组段的操作语法类似于隐式循环。
  - A(3:5)=5
- 将A(3)、A(4)、A(5)的值设置为5，其它值不变。

  - A(3:)=5
- A(3)之后所有元素的值设置为5，其它值不变。

  - A(1:5:2)=/3,4,5/
- 将A(1)、A(3)、A(5)的值设置为3，4，5，其他的值不变。

  - A(1:10)=A(10:1:-1)
- 将A(1:10)翻转过来。即将A(1)设为原来的A(10),A(2)设为原来的A(9),依此类推。

  - A(:)=B(:,2)
- 其中，假设A和B分别声明为INTEGER A(5)、INTEGER B(5,2),这里将二维数组B第二列的5个元素赋值给一维数组A的5个元素

之陳印斌

动态数组的使用

- 动态数组的使用一般要经历三个步骤：
  - 声明动态数组。规定数组的维数，但不给出维上的大小和上下界（称为延迟形状数组）。如：
    REAL,DIMENSION(:),ALLOCATABLE::X
    - **注意这个是声明语句**
  - 给动态数组分配内存。如：
    ALLOCATE(X(N))
  - 将分配的内存释放掉。如：
    DEALLOCATE(X)

之陳印斌

判断对错

- **Real,allocatable ::Mark(:)**
- Read*,n
- **Allocate(Mark(n))**
- **!Allocate(Mark)**
- **!Allocate(n)**
- **!Allocatable(Mark(n))**

- …
- **Deallocate(Mark)**
- **!Deallocate(n)**
- **!Deallocate(Mark(n))**

之陳印斌

主要内容

- Fortran基础知识
- 变量类型与表达式
- 例程和模块
- 控制结构
- 数组
- **派生类型和指针**
- 输入输出及文件操作

之陳印斌

Pointer

- 指针是一种“**间接使用数据**”的方法。指针变量用来保存一个“内存地址”。当程序要读写指针时，实际上会经过两个步骤：
  - 取出指针中所保存的内存位置；
  - 到这个内存位置读写数据。
- 指针变量中所保存的内存地址来源可以有两种：
  - 记录其它非指针变量的内存位置；
  - 程序执行中动态配置一块内存。

之陳印斌

使用指针

- **指针记录其它非指针变量的内存位置**
- 将指针指向已声明的内存地址，则可以利用指针间接地使用内存地址
- **每一种指针变量都占用相同的内存空间。因为指针变量实际上是用来记录内存地址**
- 程序说明

  integer, pointer :: p → 声明**p为**一个指针，其所指向的内存的数据类型为整形

  integer, target :: a → 声明**a为一个**可以当成目标的变量

  p => a → 将指针**p**指向变量**a**
- **以现在的32位计算机来说，记录一个内存地址，固定需要使用32bits=4bytes的空间**

之陳印斌

派生类型成员的引用

- 派生类型成员可以和同类型的变量一样使用，**引用派生类型成员时，须使用成员操作符“%”**
  - Student1%Birthdate = 461211
  - Class%Gender = .TRUE.
- **同一派生类型的两个变量可以相互赋值：**
  - Student2 = student1

之陳印斌

主要内容

- Fortran基础知识
- 变量类型与表达式
- 例程和模块
- 控制结构
- 数组
- 派生类型和指针
- **输入输出及文件操作**

之陳印斌

编辑符——I

- 使用I编辑符，一般形式Iw
  - **‘w’规定了输出的宽度，包括前导的负号所占据的一列**
- Iw.m
  - **保证有m位数字被输出，不包括负号。若不够m位，前面填0**
- 二进制Bw、八进制Ow、十六进制Zw
  - 同样可以规定最小的位数m

之陳印斌

编辑符——F

- F编辑符（Fixed point）：Fw.d
  - w: 总的输出宽度，包括符号和小数点
  - d：小数点以后的位数　f8.2
- **输入时，若字符串包含小数点，则d被忽略**
  - f8.2　1.2345仍然是1.2345
- **输入时，若字符串不包含小数点，则最右边的d位为小数部分**
  - F8.2　12345被读做 123.45

之陳印斌

写出下面程序运行的结果

```
a=123.454
b=35.368
c=123.4568
PRINT 10,A,B,C
10  FORMAT(1X,'A=',F6.3,2X,'B=',F5.2,2X,'C=',F7.3)
20  FORMAT(1X,'A=',F6.2,2X,'B=',F6.2,2X,'C=',F8.5)
30  FORMAT(1X,'A=',F8.3,2X,'B=',F6.3,2X,'C=',F8.4)
END
```

- A=****** B=35.37 C=123.457
- A=123.45 B= 35.37 C=********
- A= 123.454 B=35.368 C=123.4568

之陳印斌

编辑符——E

- E编辑符
- Ew.d、 Ew.dEe
  - w: 总的输出宽度，包括符号，小数点和指数部分
  - d：小数点以后的位数
  - **基数的绝对值小于1**
  - E10.4　　1.234E+23　->　0.1234E+24
  - e：限定指数部分的位数，不足部分填充0
  - E11.4E3　1.234E+23　->　0.1234E+024

之陳印斌

READ

- **如果输入数据多于变量个数，多余变量不起作用**
- **如果输入数据少于变量个数，fortran将把没有输入数据的数值型变量值设为0（0.0），**字符型变量设为依赖于特定系统的字符串。
- 若字符串不含空格，可以不用引号界定，否则必须加引号
- **如果有多个READ语句出现，每一个READ语句必须从一个新的输入行开始**

之陳印斌

OPEN语句

- 要对文件中的数据进行操作，首先要打开(OPEN)文件，建立文件与程序之间的连接。OPEN语句的一般形式为：
  - OPEN([UNIT=]**u,speclist**)
- 当中的u为单元号，假如不使用有名参数(UNIT=),**单元号必须出现在开头**。Speclist为格式规定列表，其中的项大多是可选的。

之陳印斌

格式规定列表

- FILE参数指文件名，假如FILE被省略，则必须提供STATUS参数，其值为SCRATCH。
- STATUS参数
  - 若规定为**SCRATCH**，一个临时文件将被创建，当文件关闭或程序结束时，临时文件随即消失；
  - 若规定为**NEW**，不能有同名文件存在；
  - 若规定为**OLD**，同名文件必须存在；
  - 若规定为**REPLACE**,文件不存在时创建新文件，文件存在时其内容将被替换；
  - 若规定为**UNKNOWN**,表示文件可以是已存在的或不存在的。
- ACCESS参数规定了文件读取模式，顺序文件缺省为SEQUENTIAL。
- FORM参数规定了是否按格式读取，顺序文件缺省为FORMATTED。
- POSITION参数代表文件指针位置，若规定为APPEND，新的数据将被追加到文件尾。

之陳印斌

CLOSE语句

- 通常，CLOSE语句和OPEN语句配套使用，它使文件和程序断开连接。其一般形式为：
- CLOSE([UNIT=]u [,STATUS=st])
- 其中，u代表要关闭文件的单元号。STATUS可选参数可以规定为KEEP或DELETE，缺省为KEEP，即关闭后文件中的数据被保留；若规定为DELETE，关闭时文件中的数据被擦除；若文件为临时文件，则STATUS只能是DELETE(缺省)。当程序正常结束时，不管是否执行了CLOSE语句，所有打开的文件都自动被关闭。

之陳印斌

例子

- open (unit = 6, file = "filename",status = 'unknown',form = 'formatted')
- Write(6,999) a,b,c
- Print 999, a,b,c
- 999 format(1x, I4,F8.2,E10.4)
- close (6)

之陳印斌

题型

- 填空（2*10=20）
- 判断正误（2*5=10）
- 选择（2*10=20）
- 读程序，回答问题（10*2=20）
- 程序改错（10*1=10）
- 编程序（20*1=20）

之陳印斌

例题：因式分解

```
!输入一个自然数，进行因子分解并输出结果，例如24=1×2×2×2×3。

program decompose

      integer number,i,d

      write (*, '(A)', advance='no') 'please input a number : '
      read*, number

      write(*, 10, advance='no') number,1    ！先输出 number=1，不换行
      do i = 2, number
            do while (mod(number,i)==0)
                  number = number / i
                  write(*, 20, advance='no'), i    !能整除的话不换行输出×i
            end do
      end do

      write(*, *)                    !最后换行使结果美观

10 format (I5,' = ', I2)
20 format (' ×', I2)

end program
```

之陳印斌

例题2：哥德巴赫猜想

```
program check
integer n,i,a,b,c,d,count

do n = 6, 100, 2
      count = 0
      do i = 1, 100, 2                    !最小的奇素数是3
            if(n >= 2*i+1) then
            a = 2*i+1                          !把x分解为两个奇数之和a+b
            b = n-a
            c = 2                              !c,d为循环变量
            d = 2

            do while( c <= a .and. mod(a, c) /= 0 )
                  c = c + 1
            end do

            do while( c <= b .and. mod(b, d) /= 0 )
                  d = d + 1
            end do

            if(c == a .and. b == d) then
                  print*, n, '=', a, '+', b
                  count = count + 1
            end if
            end if
      end do
      print*, n, '可以分解的等式个数为', count
      print*, '----------------------------------------'
end do

end program
```

之陳印斌