Fortran

Ch1. 程序基础.

✓ 不分大小写. "&" 连接下一行.

数: Integer (-2147483648 ~ 2147483647)
real. complex. Character, logical
/ / 赋值.

优先级: ( ) > ** > *./ > +.-

内部函数: abs, exp. sin. cos, asin. acos, (求PI = acos(-1.0))
tan. atan, log, log10, int(取整)
(log = ln, log10 = lg)
mod(X1,X2), sign(X1,X2) = |X1| * sgn(X2)
1, 0, -1 : sign(1, x)
max. min (一系列数)

✓ 字符串变量 (int → character, 内部文件的方式)
子串: xx(m:n) 从m ~ n (包含), 不可互相覆盖
连接: //
长度: Len , LGT. LGE, LLT. LE LLE.
INDEX (子串位置), Char(I), IChar(I)
Len_Trim (尾空格 rm)
trim 去尾空格 ; adjustL 或 adjustR

✓ 逻: .true. 或 .False.
✓ Implicit none (类型 > Implicit 语句 > I-N规则)
~常数: parameter
✓ 等价: Equivalence → save
✓ 自定义类型: Type :: xx
End Type xx
✓ 输入 Read
输出 Write
~ type1%value (取type类型变量 type1 的变量 value)

Ch2. 流程&逻辑

✓ 顺序
✓ 选择. ~ (比较) ~ 关系运算. .ne. ⟺ /= ; abs( ) < 1.0E-6
✓ 循环.
✓ 逻辑表达式: .and. .or. .not. .eqv. .neqv.

if 语句. (单, 双, 多分支.) → else 去 then
✓ Select case(x)
case(v1) ← int/character
…
case default — case(:-1) 即 ≤ -1
End select

Ch3. 循环结构:

GOTO, DO (并行: share & private)
{ Cycle. 跳过当前.
Exit 跳过当前及以后 (退出).

✓ 随机数: call random_seed 只使用一次
{ DO: 循环次数已知
DO WHILE: 次数未知, 要退出循环.

✓ Ch4. 数组: 括号 (一般从1开始)
先说明后使用: 说明语句应在可执行语句之前 → 可调整大小的数组

a(1, 2)
先表 → 行 列

输入&输出: 数组名. do 循环. 隐含do循环. ((g(i,j), j=1,3), i=1,2) ; data 语句. data x.y/…, …/
整体
a/10*-1.0/ 或 a=(/…, …/) (列表) 赋值.

✓ 可变大小的数组 (allocatable)
& deallocatable.

✓ 数组整体运算. A(1:5:2) = 3
⟺ A(1)=A(3)=A(5)=3.

⚠ 内部基本函数: SQRT, MATMUL(A,B) , DOT_PRODUCT(A,B) (点乘)
A(n,m) * B(m,k) ⇒ C(n,k)
A(m) * B(m,k) ⇒ C(k)
A(n,m) * B(m) ⇒ C(n).

✓ 求和: SUM(A[, DIM][, MASK]) , PRODUCT. MAXVAL. MINVAL
(筛选…)

逻辑运算: All(mask[, dim]), Any: 任意, Count: 数目.
(全)

数组大小: size , shape (n维 → 一维 大小为n数组), Lbound. Ubound

数组合并: Merge(…, …, MASK) ; pack (A, mask[, vec])

数组形状: spread(A, dim, ncopies), 次数. 1: 沿第1维; 2 沿第2维.
从一维数组中获取, 不够补0/vec对应位置

transpose: 转置; eoshift, cshift (末端移动, 循环替换)

定位函数: maxloc (A[, dim][, mask])

数组操作语句: where 语句, where 构造 (where - elsewhere - end where)
forall (…[, mask]) …语句 & 构造: forall - end forall

Ch5 子程序.

5.1 语句函数, 在implicit none之前, 虚参可以同名. (在本程序单元).

5.2 函数子程序 function … (内部说明类型)

千 百 十 个 ← mod 10 .
int(/1000) /100 后 mod 10 /10 后 mod 10

5.3 子例行程序: subroutine. "return"

5.4 数据传递: 字符串等赋值 (已设置好).
子程序: 可调数组 < 内在函数: intrinsic
外在(自己设计): external
(*) 作为虚参 —对应: *10 语句
✓ 虚参属性: 全局: save (Static 一次)
平均化: intent(in) 只读不改变数值
(out) 输出 ; (inout) 可读可写.
✓ 不定个数传参: optional 命令 (局部). if(present(xx)) then
else
end if
→ 接口 interface.

5.5 递归调用 (recursive) — 注意口.
recursive function 名(表) result(中间量)
…
end. ↑ 此处赋给中间变量.
或: recursive subroutine 名(表)
call 名(表)
end

5.6 共用存储单元 & 数据块子程序
equivalence (表1),(表2) … → 同一程序单元.
(数组) (变量)
公用: ①虚实参; ② 公用数据块, ③全局变量.
不同程序单元 < Common xx, yy
function 中 common aa, bb (类型同, 个数可不同, 按顺序对应)
→ 有名公用区 common /ARY1/ , …
数据块子程序: Block DATA.

5.7 内部子程序: contains (subroutine 中不可嵌 contains subroutine END 但可调用)

PB18071498 刘炜昊

Ch6 格式化输入输出

隐含设备 print *(...)(格式) 变量... ~ read *(*)...

write (设备号, 格式). 输出项. ~ read (*,*)

格式(语句). n (行号) format(...) ~ 非执行语句.

编辑描述符:

可重复: 整型: I, B, O, Z
十 二 八 十六
rIw.x } 一样.
个数 宽度 位数

实型: F, E, EN, ES, G → 1~1000, 指数可被3整除 / 视E(综合) / 非指数部分: 1~10
rFw.d 指数 (指数位: 4位 E+01, E-02) 0.7E..
个数 宽度 小数位数 (输入时自带小数点优先)

复型: 2F... (两个数).

逻辑型: rLw ⊔T/F + 任意字符
输出: T/F. 左补 w-1 个空格.

字符型: rAw 可省 → len. (w 省略后不能 →默认;
符有前缀字符串时. w<L: 选从左对齐).
→若加了, 也作为字符串中一个字符.
出: w>L: 右对齐. 左补 w-L 空格
w<L: 将最左边 w 个字符输出.

非重复: 不作为输入输出项对应.
X 编辑符: nX (n 个空格).
H 编辑符: nH h1...hn (n 个字符). 将其后的内容字符输出.
撇号: 'C:' → 直接输出.
斜杠: 换行.

多于变量数. 多出不用. 少于: 未重新开记录.

{ 隐含多次循环 → 重复 r 次: 单记录
{ 显式 do 循环重复: 多记录.

Ch7. 文件

7.1 概念: 外部 / 内部 } 存取方式 < 直接 → recl 指定长度 / 顺序

结构: 有/无格式 & 二进制)
(字符, ASCII码) → 乱码 → 乱码.
(formatted, unformatted, binary)
格式化 → 只可有格式顺序存取. 计较号!
Form 有默认有格式

7.2 打开: OPEN(unit=10, File='...', ACCESS='...', sequential (有默认)
当前目录下. Status='replace'

close(10).

7.3 读写: 有格式顺序存取. read & write (含fmt 说明) 指定 or not.
(P418)
rewind: 退回到文件头.
有格式直接存取. 加上控制项 REC=rec → 视作为一行记录.
△ 无格式顺序存取 WRITE(10)~ (一记录一行).
~ 无格式直接存取, 二进制(binary) 顺序/直接存取.
内部文件: ASCII码 → integer, locical, charater.
(内部文件) ↑read / write (外部变量)

7.4 其他操作: rewind: 顺序+直接; backspace (回退1记录)
print, endfile; inquire (无格式, 直接). (P447)
EOF: 文件操作函数值特殊(有/无).

Ch8 module & 面向对象.

8.1 结构化. 面向对象 (公开&私有).

8.2 模块化 (可继承) △ 要写 save & implicit none.
contains
subroutine 或 function → 模块函数库.
public & private 属性 → 不可直接调用.
继承方法 use a, only: vc (仅用...)
use a, aa => va (改名)

8.3 接口: interface (程序模块 ~ "使用接口") → 重载 & 自定义操作符. interface operator(+)

Ch9 并行初步. (OPENMP) — 注释后不影响串行.

!$OMP Parallel Shared( ) Private( )

!$OMP End Parallel

(内部) 线程总数 & 线程编号 OMP_Get_thread_num 输出: 互相穿插
OMP_Get_num_threads ↑ 各自编号.

任务共享: !OMP DO/END DO (默认均匀分配)
no wait → 不等待 (默认等) ~ 无关时才可省.
Schedule (Dynamic, Chunk) 动态分, 每次分 chunk 个.
互相依赖 ⇒ 空间换效率. 再多用一个数组可.

!$OMP sections/END Sections (手动分配)
!$OMP Section
--
!$OMP Section

!$OMP Single/END Single (单线程执行, 谁遇谁执行, 默认不等)

!$OMP Workshare/END Workshare (向量数组计算)

合并. !$OMP Parallel Do clause...
...
!$OMP END Parallel Do. (或 Sections 或 Workshare).

同步指令: !$OMP MASTER/END ~ (仅主线程执行). 不等待, 不同步.
!$OMP critical/END ~ (同一时间仅一个线程工作, 如写入文件). 不同步. 依次执行.
!$OMP Barrier: 当所有线程到达.
!$OMP Atomic: 所有线程依次执行.
!$OMP Flush: 数据更新 → 使所有线程看到的一样.
!$OMP Ordered/End ~: 以顺序执行.

属性: private, shared, Threadprivate
!$OMP ~, 主要用于全局变量 (保证在各线程中独立)

other: !$OMP Parallel if (N>1000) ; !$OMP Parallel num_threads(4)
!$omp Do. 开4线程.

△ Ch10. 算法实例. (并行. 模块. 结构. 刷图).

① 积分: 矩形法, 梯形法, Simpson 法.
P551 P553 P556

② 解方程: 迭代法 P559, 牛顿迭代法 P563, 二分法 P566, 弦截法. (P569)
(直代: 迭代格式!) (知导数!) (异号区间!) (异号区间! 更快!)

③ 求极小值: 0.618 — 黄金分割法. P572-573

④ 插值近似: 精确 & 平滑度 (分段) — 多项式插值. ~ Lagrange 插值. → 二维 (~P583)

⑤ 一阶 ODE 组的求解: RK4 (P587)

⑥ PDE: 对流. 扩散. 对流扩散 & 场. → 有限差分近似 - (一阶!)
· 对流 ~ 迎风格式. (柯朗数 CFL, 0.8 左右, ~ 尽可能小)
(P594) $\Delta t = CFL \times \frac{dx}{|a|}$.
i=1 → (2: NX+1) ; 0: → (0: NX-1).
· 扩散 ~ 显格式 (P609).

PB18071498 刘炜昊