# Fortran 学习笔记

一.基本操作

1.Visual Studio

（1）安装 Visual Studio + oneapi（网上有教程，安装 2022 版紫色那一款即可）

（2）基本操作

文件->新建->项目->Fortran->Main Programe Code

文件->新建->文件->Inter(R) Fortran-> Fortran Free-form File (.f90)

*推荐第一种方式

2.linux 系统（服务器/大型机）

• 创建并打开文件

vi xxx.f90

• 显示文件内容

cat xxx.f90

• 编译程序

gfortran xxx.f90

会发现生产一个 a.out 的文件，说明就已经编译成功了；然后输入命令 ./a.out 即可运行这个 Fortran 程序：example.f90

二.Fortran 基本程序结构

```
program 程序名

implicit none  !（不允许隐式变量声明，必加上）

!主体内容

pause/stop  !停止语句

end program 程序名
```

Fortran 的一些特点：

1.不区分大小写，A 和 a 是一样的

2.对于缩进没有强制要求，但是可以通过缩进改善可读性

三、Fortran 语言基础

1.基本数据类型

*赋值声明：所有常量、变量应声明在开头，之后就不能声明了

（1）常量

```
real, parameter :: a = 5.35E5  !科学计数法：5.35E5 表示 5.35×10^5

real, parameter :: pi = 3.1415927
或者
real pi
parameter(pi=3.1415927)
```

## 2.变量

参数：kind 值=取值范围，integer(1)=-128~1
integer(2)=-32768~32767…
::符号：赋初值　　变量名只能以字母开头

• integer 整型变量
integer(kind=2) a,b,c,d　!声明 kind 值为 2 的 4 个整型变量
integer(1) e　!声明 kind 值为 1 的一个整型变量
integer ::g = 123　!声明 kind 值为 4（默认）的 1 个整型变量且初值为 123

• real 实型变量（浮点数）
real ::g = 1.23
real f
f = 100.0

• character 字符型变量
character(len=10) e,f,g　!声明长度为 10 的 3 个字符型变量
character *6 str　!声明长度为 6 的 1 个字符型变量
character ::a = ”A”
add = string1//string2　!链接两个字符串
string1(6:14) = ”afternoon”　!重新设置第六个字符和第 14 个字符之间的字符

• complex 复型变量（复数）
complex ::g = (3,4)

• logical 逻辑型变量
logical a
a = .true/false

## 3.运算符

（1）算术运算符
加+　减-　乘*　除/　乘方（计算幂）**
（2）关系运算符
相等==　不等/=　大于>　小于<　大于等于>=　小于等于<=
（3）逻辑运算符
与.and.　或.or.　非.not.

## 4.程序控制语句

• pause 语句
暂停程序的运行，当需要恢复程序的运行时，只需要按回车键即可
一般形式为 pause[n]，[n]为执行 pause 时输出的内容（可有可无）
pause　“abcd”　!暂停的同时，输出”abcd”
• stop 语句
停止程序的运行，一般形式为 stop[n]
• end 语句
每个程序单元必有一个 end 语句在该程序单元的最后一行
子程序中的 end 语句是一个程序块的结束标志，在之前使用 return
主程序中的 end 语句表示整个程序中的终止执行语句，在之前使用 stop/pause

## 5.数组

• 用类型说明语句定义数组

```
integer a(-5:5),b(20)   !a 数组下标从-5 到 5，一共 11 个元素；b 数组下标从 1 到 20，一共 20 个元素
character *8 name(50)   !name 数组一共有 50 个数组元素，每个元素可以存放长度为 8 的字符串
integer :: a(5) = (/1,2,3,4,5/)  !a(1)=1 a(2)=2 a(3)=2 a(4)=2 a(5)=5
integer :: a(5) = (/(i,i=1,5)/)   !a(1)=1 a(2)=2 a(3)=3 a(4)=4 a(5)=5
```

• 用 dimension 语句定义数组

```
dimension a(15:5),b(3,4)
integer a
real b
```

• 二维数组

```
integer :: a(3,3)
integer :: a(2,2)=(/1,2,3,4/)   !a(1,1)=1 a(2,1)=2 a(1,2)=3 a(2,2)=4 ，数组的储存顺序为按列储存，与 c 语言相反
```

• 数组的输入

```
do i =1,10
    real(*,*) a(i)
end do
```

• data 语句赋值

```
data a/1,2,3,4,5/    !可放在说明语句之后，end 语句之前的任何位置
date (num(i),i=1,10)/10*0/   !给 num 数组前 10 个数都赋值为 0
```

• 数组的输出

```
do i=1,10,2
    write(*,200) a(i)
end do
200 format(1X,I4)
```

• 二维数组的输入输出

```
do i=1,3   !按行输入，一共 3 行 2 列
    do j=1,2
        read *,w(i,j)
    end do
end do
```

• 数组运算

```
a=b+c              !实现矩阵相加
a=b-c              !实现矩阵相减
a=matmul(b,c)      !实现矩阵相乘
d(3:5)=5           !对数组部分操作  d(3)=d(4)=d(5)=5
```

• 动态数组（开始时没有定义元素个数的数组）

```
integer,allocatable::vector(:),matrix(:,:)   !定义一维整型动态数组
read *,n   !从键盘上输入数组的大小
allocate(vector(n))    !根据读入的 n 为数组分配内存空间
deallocate(vector)     !释放动态数组剩余内存
```

6.系统内部函数

```
abs(x)     !求 x 的绝对值
exp(x)     !求指数函数 e^x
sin(x)     !求正弦函数 sin(x)，其中 x 的单位为弧度
asin(x)    !求 arcsin(x)
log(x)     !求 In(x)
int(x)     !求 x 的整数部分，不四舍五入
max(x1,x2,…)     !求 x1,x2,…中的最大值
real(x)    !把整型变量 x 转换为实型
mod(x1,x2)    !求 x1 除以 x2 的余数
```

7.输入输出语句

• read 表控输入语句　　*=表控输入/输出，可指定类型

read *,a,b　!如果 a 为实型，m 为整型。从键盘输入 3.7,-24 或者 3.7　-24 即可使 a=3.7,b=-24

使用多个 read 语句时，每个 read 语句都是从一个新的输入行开始读数的

• write 表控输出语句

write(*,*) "hello,world"　!输出”hello,world”

• print 表控输出语句

print *,a　!输出变量 a

• 整数的有格式输入

I（大写 i）编辑符专门用于整型数据的格式说明，其格式为 Iw，w 表示字段总宽度

```
integer i,j
read(*,"(I4,I5)"),i,j
print"(1X,'i=',I5,j=',I5)",i,j
```

输入 123412345，则会输出 i=1234 j=12345，其中 1X 代表 1 个空格

可以使用,间隔，表示强行终止前一数据开始下一数据；此时的空格不能作为间隔符号

• 浮点数的有格式输入

Fw,d 表示浮点数的有格式输入，w 表示字段总宽度（包括小数点一列），d 表示小数位数

• format 格式说明语句

格式为 s format(格式说明)，其中 s 为格式说明语句的编号，以便相同格式的输入输出语句配合使用

```
  character *5 c1,c2
    read (*,10),c1,c2
10      format(A3,A4)
    print *,c1,c2
```

输入 asdfghj，输出 asd fghj，c1=asd，c2=fghj

read 和 format 中的 10 均表示语句编号

• Aw 表示字符串的有格式输入

8.选择结构

（1）if…then 结构

可使用 else、else if 等

• X 编辑符　　表示输出空格，格式为 nX

• 斜杠编辑符　表示换行

（2）select case 结构

select case 语句允许一个变量，列表进行相等测试。每个值被称为一个情况，在该变量上检查每个选择的情况 select case 。

• if 结构

```
if(条件 1) then
    if 块
else if(条件 2) then
    else if 块
else
    else 块
end if
```

• 逻辑 if 语句

```
if(条件)  语句
```

• select-case 结构

```
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
        …
end select
```

• 运算符

.and.  .or.  .not.  等

## 7.循环结构

• do 结构（列表循环）  !结构名可以不写

```
[do 结构名:]do n=1,30,1    !其中最后一个参数为步长（可有可无）
    read *,num,grade
    print *,num,grade
end do[do 结构名]
```

循环输入学生的学号和成绩并打印，重复 30 次

• do-while 结构

```
[do 结构名:]do while(逻辑表达式)
    循环体
end do[do 结构名]
```

• exit 语句

停止循环，通常和 if 语句结合使用

```
if(逻辑表达式) exit[do 结构名]
```

• cycle 语句

中止本次循环，并重新返回到 do/do while 语句开始执行

```
cycle[do 结构名]
```

• 无循环变量的 do 结构

```
[do 结构名:]do
    循环体
end do[do 结构名]
```

配合 exit 语句使用，在一些条件下可以使得循环更整洁清晰

## 8.数组

数组的索引值是从 1 开始的

9.子程序

• 子程序

```
program main
    call sub1()   !调用子程序
end program main
subroutine sub1()
    …
end subroutine sub1
```

• 函数

```
program main
    implicit none
    integer,external ::myfunc   !声明函数&类型
    integer ::a=2,b=3,c
    c = myfunc(a,b)   !调用函数
    write(*,*) c
    pause "enddddd"
end program

function myfunc(a,b)
    implicit none
    integer ::myfunc,a,b !声明返回值和函数参数类型
    myfunc = a+b
    return
end
```

• 全局变量

定义的是一块共享的内存空间，假设在子程序中使用时只用第三个全局变量，也要声明前两个

```
common a,b,c
integer a1,b1   !使用时需要数据类型声明
```

全局变量分类，可根据需要声明

```
common /group1/ a,b
common /group2/ c
```

在 block data 块中进行 common 初始化

```
block data
    implicit none
    integer a1,b1
    common a1,b1
    data a1,b1/111,222/
end block data
```

利用全局变量实现主程序/子程序之间的参数传递

```
program console2
    implicit none
    external sub
    integer a,b,c
    common /n1/a
    common /n2/b
    common /n3/c
    read *,a,b
    call sub
    write(*,*),c
    pause "enddddd"
    end program

subroutine sub
    implicit none
    integer x,y,z
    common /n1/x   !x 和 a 公用一个存储空间
    common /n2/y
    common /n3/z
    z=x*y
    end
```

• 内部子程序

subroutine 子程序名(虚参)

…

end subroutine 子程序名

• 内部函数

function 函数名(虚参)

…

end function 函数名

内部子程序/函数中 end 关键字必须包含 subroutine/function，但在函数/子程序中这是可选项

内部子程序中不能再包含内部子程序

• interface 模块

用于函数返回值为数组/指针/所调用的函数参数数目不固定等情况

## 10.结构体和指针

用于处理不同类型的多组数据

## 11.文件

• 打开文件

open(10,file='test.txt') !unit 为文件指定的 id，是一个 9-99 的正整数。10+为佳

write/read 第一个*号的位置改成文件 id，即可实现文件读写

write(10,*) "Good morning"

创建新的临时文件：

open(1, file='data1.dat', status='new')

• 关闭文件

close(10)

```
program outputdata
implicit none

    real, dimension(100) :: x, y
    real, dimension(100) :: p, q
    integer :: i

    ! data
    do i=1,100
        x(i) = i * 0.1
        y(i) = sin(x(i)) * (1-cos(x(i)/3.0))
    end do

    ! output data into a file
    open(1, file='data1.dat', status='new')
    do i=1,100
        write(1,*) x(i), y(i)
    end do
    close(1)

    ! opening the file for reading
    open (2, file='data1.dat', status='old')
    do i=1,100
        read(2,*) p(i), q(i)
    end do
    close(2)
    do i=1,100
        write(*,*) p(i), q(i)
    end do
end program outputdata
```