第五章 子程序

5.1 基本概念

基本概念 程序代码中需要重复某个功能或重复使用某一段程序代码时，可用子程序。在其它语言中，可以被理解为函数。一个程序包含一个主程序（程序入口）和若干个被按需调用的子程序。

子程序包括：函数子程序(function)、子例行程序(subroutine)、数据块子程序(block data)

分类 子程序又分为：外部子程序和内部子程序。

5.2 外部子程序

分类与描述 外部子程序包括外部函数子程序和外部子例行程序。外部子程序的定义位于调用程序单元的外部。也就是说，它位于主程序的 END 语句之外。

5.2.1 外部函数子程序

描述 由一个实现某种特定功能的子程序组成，调用程序单元调用它得到一个函数值（仅仅只有一个）。

定义形式 [类型说明] **FUNCTION** 函数名([虚参 1,虚参 2,…]) 类型说明定义了返回的函数值的类型

说明语句 变量定义

执行语句 具体的可执行代码

END [FUNCTION [函数名]] 返回值存放在函数名中

调用形式 函数名(实参 1,实参 2,…) 或者 函数名()

执行步骤

① 在调用程序单元的函数调用表达式中，计算实参值。

② 将实参值传递给对应虚参，即虚实结合。

③ 执行函数体，计算函数值，并将函数值赋给函数名。

④ 执行 END 语句，将函数值带回（即返回）调用程序单元的函数调用表达式。

注意

① 实参的个数、类型和位置顺序必须与所调用的，函数子程序对应的虚参一致。

② 实参名称与虚参名称可以相同，也可以不同。

③ 实参与虚参主要由位置顺序建立其对应关系。

例题：已知上海、南京、武汉三地的气温，编制外部函数子程序计算三地的平均气温

```
program $251113    主程序
    implicit none
    external average
    real a,b,c,average
    read *, a,b,c
    print *, average(a,b,c)
end program $251113

real function average(x,y,z)    子程序部分
    implicit none
    real x,y,z,average          对三个虚参进行数据类型定义
    average = (x+y+z)/3         注意：这里 average 可以是个变量
end function average
```

external 语句说明 average 是一个外部子程序，可以省略。

主程序中也必须对 average 进行数据定义。

5.2.2 外部子例行程序

描述 子例行程序不仅可求一个值，还可求多个值或不求值而执行某种操作，因此具有更广泛的用途。

定义形式 **SUBROUTINE** 子例行程序名([虚参 1,虚参 2,…]) 不存在类型定义了，程序名不承担变量作用

说明语句 变量定义

执行语句 具体的可执行代码

END [SUBROUTINE [子例行程序名]]

调用形式 外部子例行程序的调用必须由调用语句（即 CALL 语句）实现。

一般形式为：**CALL** 子例行程序名(实参 1,实参 2,…)

执行步骤

① 在调用程序单元中，计算实参值。

② 将实参值传递给对应虚参，即虚实结合。

③ 执行子程序体，实现其功能。

④ 执行 END 语句，将返回值带回给调用程序单元。

例题：已知上海、南京、武汉三地的气温，编制外部子例行程序计算三地的平均气温

```
program $251113    主程序
    implicit none                    subroutine average(x,y,z,ave)
    external average                     implicit none
    real a,b,c,temp                      real x,y,z,ave
    read *, a,b,c                        ave = (x+y+z)/3
    call average(a,b,c,temp)         end subroutine average
    print *, temp
end program $251113              按照位置传递数据，这里把 ave→temp。
```

两者区别

① 函数子程序会返回一个数值，并对储存返回值的函数名要有类型声明；子例行程序可以返回 1 个、多个或不返回。

② 函数子程序调用时不用 call，但调用前要对函数子程序名声明；子例行程序必须用 call 命令调用

③ 函数子程序体中，由于用函数名存放函数值，故函数名具有类型；而由于子例行程序的计算结果均存放在虚参表中，故子例行程序名既不存放数值也没有类型。

5.3 内部子程序

分类与描述 内部子程序包括内部函数子程序和内部子例行程序。内部子程序的定义一般位于调用程序单元中 CONTAINS 语句之后，只有包含内部子程序的程序单元才能调用它们。

5.3.1 内部函数子程序

定义形式 [类型说明] FUNCTION 函数名([虚参 1,虚参 2,…])

说明语句

执行语句

END FUNCTION [函数名]

调用形式 函数名(实参 1,实参 2,…) 或者 函数名() 与外部函数子程序相同

例题：已知上海、南京、武汉三地的气温，编制内部函数子程序计算三地的平均气温

```
program
implicit none                        因为是内部子程序，无需 external 语句
    real a,b,c                       real 中也无需定义 average 的类型
    read *, a,b,c
```

```
        print *, average(a,b,c)
        contains                                  由该行定义其之下的语句都属于子程序部分
        real function average(x,y,z)
            implicit none
            real x,y,z,average                    必须要对 average 进行类型声明
            average = (x,y,z)/3
        end function average                      为了和主程序区别，end 后必须增加 function。
    end program $251113
```

5.3.2 内部子例行程序

定义形式 SUBROUTINE 子例行程序名([虚参 1,虚参 2,…])
说明语句
执行语句
END SUBROUTINE [子例行程序名]

调用形式 CALL 子例行程序名(实参 1,实参 2,…) 与外部函数子程序相同

例题：已知上海、南京、武汉三地的气温，编制内部子例行程序计算三地的平均气温

```
    program
    implicit none
        real a,b,c,ave                            这里需要定义 ave 的类型
        read *, a,b,c
        call average(a,b,c,ave)
        print *, ave
        contains                                  由该行定义其之下的语句都属于子程序部分
        subroutine average(x,y,z,ave)
            implicit none
            real x,y,z,ave
            ave = (x,y,z)/3
        end subroutine average                    为了和主程序区别，end 后必须增加 function。
    end program $251113
```

注意
① 内部子程序的定义应当位于调用程序单元中的 CONTAINS 语句之后。
② 内部子程序中不能再包含内部子程序。
③ END 语句中的关键字 FUNCTION (或 SUBROUTINE)在外部子程序中是可选项，但在内部子程序中是必选项。

5.4 实参和虚参之间的数据传递

虚实结合主要方式 调用子程序时，用实参取代虚参的过程,是不同程序单元数据传递的主要方式。
① **地址传递**：将实参的地址传递给虚参，虚参和实参拥有相同的内存地址，虚参值的变化会改变实参值，如 Fortran 语言。
② **值传递**：将实参的值传递给虚参，虚实结合后虚参不会改变实参的值，如 C 语言。

虚参类型 变量作为虚参、数组名作为虚参、子程序名作为虚参、星号（*）作为虚参

5.4.1 变量作为虚参

概述 子程序用变量作虚参时，对应的实参可以是**同类型**的常量、表达式、变量或数组元素。

常量/表达式 实参是常量或表达式时，采用值传递方式，将实参常量、表达式之值传递给虚参变量。

变量/数组　实参是变量或数组元素时，采用地址传递方式，将实参变量或数组元素的地址传递给虚参变量。调用之前，虚参没有值。调用时，虚参的值就是实参的值。

示例

```
program $251113
    implicit none
    external sub
    integer :: a(5)=(/1,2,3,4,5/), b=8      创建了一个有五个元素的数组
    call sub(a(3),b)                        a(3),b 作为实参往虚参传递
end program $251113
subroutine sub(x,y)
    implicit none
    integer :: x,y
    x=x+3                                   虚参改变，实参一并改变，此处改变了 a(3)的值
    y=y+4                                   此处改变了 b 的值
end subroutine sub
```

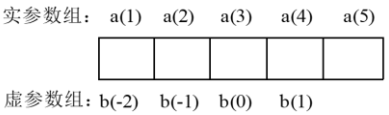
5.4.2 数组名作为虚参

概述　子程序用数组名作虚参时，对应实参可以是**同类型**的数组名或数组元素。实参数组和虚参数组可以有不同的维数和不同的长度。这时实参和虚参之间以地址传递方式实现数据传递。

5.4.2.1 实参是相同维数的数组名

情况　实参与虚参维度相同，但个数可以不同 call sub(a)。

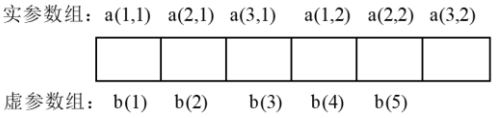
传递规则　虚参数组与实参数组的第一个数组元素首先结合，虚、实数组中的其余元素按其在内存的排列顺序（按列存放）依次结合。



5.4.2.2 实参是不同维数的数组名

情况　实参与虚参维度不同，实参二维，虚参一维。

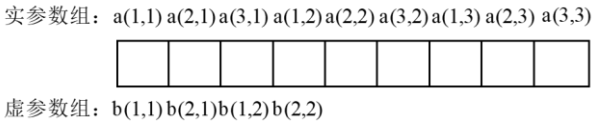
传递规则　主程序调用子程序时，二维实参数组的第一个元素与一维虚参数组的第一个元素首先结合，其余元素按其在内存的排列顺序依次结合（注意列优先规则）。



5.4.2.3 实参是维数相同，长度不同的数组名

情况　实参二维有 9 个数据，虚参二维有 4 个数据。

传递规则　实参数组的第一个元素与虚参数组的第一个元素首先结合，其余元素按其在内存的排列顺序依次结合（注意两个数组都遵循列优先规则）。

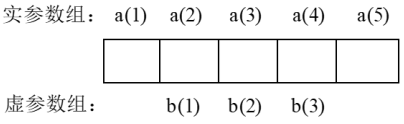


特别注意　虚参数组元素的个数必须小于等于实参数组元素的个数，否则，多余的虚参数组元素会导致子程序数据处理出错。传入的数据只能更多，不能更少。

5.4.2.4 实参是相同维数的数组元素

情况　右图所示，我们从 a(2)开始传入数据 call sub(a(2))。

传递规则　将虚参数组第一个数组元素与实参数组元素首先结合，虚参数组的其余元素与实参数组的后续元素按其内存排列顺序依次结合。



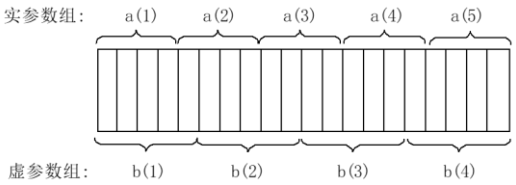
5.4.2.5 实参是不同维数的数组元素

传递规则　将虚参数组第一个数组元素与实参数组元素首先结合，虚参数组的其余元素与实参数组的后续元素按其在内存的排列顺序依次结合。原则和 5.4.2.2 完全一致。

特别注意　虚实结合后，虚参数组的所有元素均应在实参数组的范围之内，否则，由于虚实结合的错误，导致子程序数据处理出错。也就是说实参必须大于等于虚参个数。

5.4.2.6 虚参是字符型数组，实参也是字符型数组

情况　此处 a 为 character *4 a(5)　有 5 个长度为 4 的元素
此处 b 为 character *5 b(4)　有 4 个长度为 5 的元素



传递规则　虚参数组和实参数组不是按数组元素的顺序一一对应结合，而是**按字符位置**一一对应结合。虚实结合时，不仅允许虚实数组的维数不同，同时还允许虚、实数组元素字符的长度可以不同。

5.4.2.7 虚参是可调数组

传递规则　所谓可调数组，是指该数组的维界表达式（即数组的界）是个尚未确定值的整型变量，调用子程序时，该整型变量由调用程序单元给定。

示例

```
subroutine sub2(b,n,m)
    implicit none
    integer i,j,n,m,s
    integer b(n,m)          调用主程序时可以使用  call sub2(a,5,5)
```