接口界面块：Interface
功能详解

说明：子程序可看做无返回值的函数，为了方便叙述，如未特别说明，文中将函数(function)和子程序(subroutine)统称为函数。

函数的接口信息用于告诉编译器应该如何正确调用该函数，它包括参数和返回值的数量、类型等信息。因此每个函数都必须具有相应的接口信息，缺省情况具有隐式声明，而使用 interface 则可显式声明函数的接口信息。Interface 的主要功能：

1、明确外部函数(external procedure) 或虚函数(dummy procedure)的接口信息，包括：输入输出参数的类型和属性、返回值类型、函数属性；

2、定义通用过程名，即函数重载(overload)；

3、操作符(+,-,*,/,et al)和赋值符(=)重载。

下面将分别对以上三种功能进行说明。

1、接口界面块

内部函数(contains)、模块(module)中的函数，以及 Fortran 标准函数 (如：sind、abs 等) 均自动包含显式接口，不需要也不能再次声明接口信息，因此上述情况不在讨论之中。我们建议将外部函数封装在 module 中使用。

外部函数缺省具有隐式接口，对一些常规函数，用户不必显示声明其接口信息，编译器也能正确识别。但当外部函数的形参具有 ALLOCATABLE, ASYNCHRONOUS, OPTIONAL, POINTER, TARGET, VALUE, VOLATILE 属性时，必须显示声明接口信息。下列情况之一也必须使用接口界面块：

- 外部函数返回值为指针、数组或可变长度字符串；
- 形参为假定形状数组(assumed-shape arrays)；
- 参数个数不确定或包含可选参数；
- 改变参数传递顺序；
- 子程序中扩展了赋值号的使用范围。

接口界面块的使用较为简单，在接口界面块(interface-end interface)之间写入函数形参(包括返回值)类型说明语句，即去掉所有执行语句以及局部变量声明语句后的剩余部分。下面的例子给出了函数返回数组时以及具有可选参数时使用 interface 的例子：

```
program Fcode_cn
  integer::i=0, j=1, k=2, m(2)
```

```
  interface  ! 接口块
    subroutine sub1(i,j,k)
      integer,optional::k
      integer i,j
    end subroutine

    function func1(j,k)
      integer j, k
      integer func1(2)
    end function
  end interface

  m = func1(j,k)
  print*, m  ! m=3,-1

  call sub1(i,j,k)
  print*, i  ! i=3

  call sub1(i,j)
  print*, i  ! i=1

  pause
end

function func1(j,k)
  integer j, k
  integer func1(2)
  func1(1) = j + k
  func1(2) = j - k
end function

subroutine sub1(i,j,k)
  integer,optional::k
  integer i,j
  if( present(k) ) then
    i = j + k
```

```
  else
    i = j
  end if
end subroutine
```

2、函数重载

某些情况下，我们需要对不同类型或不同数量的参数做相似或相同的操作，由于参数类型、数量不同，我们需要编写不同的函数来处理。比如求绝对值，如果参数是 4 字节整数，我们需要调用 iabs 函数；如果参数是 4 字节或 8 字节实数，我们需要分别调用 abs 或 dabs 函数。由于需要记住多个功能相同或相近的函数，增加了我们使用这些函数的难度，同时也增加了出错的可能性，比如将实数传递给 iabs 函数。

上述函数的功能相同，只是参数类型或个数不同，那么可否使用同一个函数名来执行它们呢？当然可以，这就是函数重载。函数重载允许通过调用通用过程名来执行特定函数。当用户调用通用过程名时，编译器首先检查传入参数的类型和数量，再调用与之匹配(类型和数量相同)的特定函数来执行具体任务。例如我们建立通用过程名 abs 来求绝对值，用户在任何情况都只需调用 abs，编译器会自动选用合适的特定函数执行对应操作：当传入参数是 4 字节整数，就调用 iabs 函数；如果是 8 字节实数，则调用 dabs 函数。

下面用求绝对值的例子说明函数重载功能。为与 Fortran 内在函数 abs 区别开来，我们在函数名后添加“_f”。

```
module abs_module
  implicit none
  interface abs_f
  module procedure abs_f, dabs_f, iabs_f
  end interface
contains
function abs_f(x)
real(4) abs_f, x
if( x<0.0_4 ) then
  abs_f = -x
else
  abs_f = x
end if
end function abs_f

function dabs_f(x)
real(8) dabs_f, x
```

```
if( x<0.0_8 ) then
  dabs_f = -x
else
  dabs_f = x
end if
end function dabs_f

function iabs_f(x)
integer(4) iabs_f, x
if( x<0 ) then
  iabs_f = -x
else
  iabs_f = x
end if
end function iabs_f
end module abs_module

program Fcode_cn
use abs_module
real(4):: x=-2.0_4
real(8):: y=-3.0_8
integer(4)::z=-4

print*, abs_f(x)
print*, dabs_f(y), abs_f(y)
print*, iabs_f(z), abs_f(z)

pause
end
```

执行结果：

2.000000

3.00000000000000 3.00000000000000

4 4

代码分析：我们使用 interface 创建了一个函数重载，其通用过程名(interface 之后的标识符)为 abs_f，包含三个特定过程 `abs_f, dabs_f, iabs_f`。需要注意，通用过程名可以与其中一个特定过程名一致，也可以不一致；但特定过程名之间必须不同。每个特定过程的形参类型和数量不能完全一致。

```
  interface abs_f
  module procedure abs_f, dabs_f, iabs_f
  end interface
```

在程序执行过程中，调用通用过程名时，编译器首先检查实参的类型和数量，并与特定过程的接口信息相匹配。如果匹配成功，则调用相应特定函数(故dabs_f(y)和abs_f(y)的结果是一致的)；否则编译器会报错。比如调用abs_f(12_8), 由于我们没有给出针对8字节整数求绝对值的特定函数，编译器则会报错。

3、操作符和赋值符(=)重载

常规运算中，我们经常用到算术操作符(+,-,*,/,**)、关系操作符(<,<=,>,>=,==,/=)以及赋值符(=)，同时也会发现其使用具有一定的局限性：只能对特定的数据类型进行运算，不能直接用于派生数据类型。比如两个时间相减，我们不能直接使用“-”进行操作，而需要编写特定的函数。针对这一问题，Fortran90引入了操作符重载功能。我们先看下面一段代码：

```
module time_class
  implicit none
  type time  !定义一个时间类结构体
    integer(1) hour, minute, second
  end type
  interface operator(-) !重载操作符-
    module procedure timeMinus
  end interface
  interface assignment(=) !重载赋值符=
    module procedure assign_time
  end interface
  interface operator(.minus.) !自定义操作符.minus.
    module procedure timeMinus
  end interface
contains
! 两个时间相减
function timeMinus(time1,time2)
type(time),intent(in)::time1, time2
type(time) timeminus
integer n
n=(time1.hour-time2.hour)*3600      +      (time1.minute-time2.minute)*60      +
(time1.second-time2.second)
if(n<0) n=n+3600*24
timeminus.second=mod(n,60)
n=n/60
timeminus.minute=mod(n,60)
timeminus.hour=n/60
end function
! 如果time类数据正确，将其赋值给res；否则输出错误提示
```

```
subroutine assign_time(res,time1)
type(time),intent(in):: time1
type(time),intent(out):: res
if(time1.hour>=0 .and. time1.hour<=23) then
  if(time1.minute>=0 .and. time1.minute<=59) then
    if(time1.second>=0 .and. time1.second<=59) then
      res.hour=time1.hour
      res.minute=time1.minute
      res.second=time1.second
      return
    end if
  end if
end if
write(*,*) "time类数据错误."
end subroutine
end module

program fcode_cn
  use time_class
  type(time) time1,time2,time3
  time1=time(2,3,58)
  time2=time(23,12,7)
  time3=time2-time1 !重载操作符-
  print*,time3
  time2=time3 .minus. time1 !自定义操作符.minus.
  print*,time2
  time1=time(25,5,6) !重载赋值符=
pause
end
```

模块 time_class 中定义了一个时间类结构体以及两个函数，函数 timeMinus 用于操作符“-”重载和自定义操作符，子程序 assign_time 用于赋值符“=”重载。注意，操作符重载只能使用 function，而赋值符重载只能使用 subroutine，自定义操作符需位于两个 dot 之间。

程序一开始使用赋值符重载功能对 time1 和 time2 进行赋值，由于数据无误，正常执行；接下来，分别使用重载操作符“-”和自定义操作符“.minus.”进行两个时间相减，得到相应结果；最后对 time1 赋值时，由于数据有误，输出错误提示。

执行结果：

21 8 9

19 4 11

Time 类数据错误.