Fortran tahák [17.10.2018 15:44:50]

Legenda: [] = nepovinné []* = nepovinné, ale doporučené

Základ

```
program nazevPrograma
   [use nazevModulu]
   [implicit none]*

   deklarace proměnných

   příkazy

   [contains
   funkce a subroutiny]

end program

[funkce a subrutiny]
```

Obecné

stop – zastaví program

! - komentář

Proměnné

integer[(p)]
real[(p)] -
complex[(p)]
(p) = precision; (4)=single precision; (8)=doubleprecision
character[(l)] - *l=délka*
logical - .true. nebo .false.
:: - pro definici při deklaraci
„*typ*“**,parameter ::** - konstanta
„*typ*“**,intent(t) ::** - pouze v subroutině; t=OUT, IN, INOUT

---

přetypování

„*integer*“ = **int**(„*real*“)
„*integer*“ = **nint**(„*real*“)
„*integer*“ = **floor**(„*real*“)
„*integer*“ = **ceiling**(„*real*“)
„*real(4)*“ = **real**(„proměnná„)
„*real(8)*“ = **dble**(„proměnná„)
„complex“ = **cmplx**(„*real*“,“*real*“)
„*real*“ = **real**(„complex“)
„*real*“ = **aimag**(„complex“)

---

logické výrazy

>=, <=, ==, /=, >, <
.and. - a zároveň
.or. - nebo
.not. - negace
.eqv. - je ekvivalentní
.neqv. - není ekvivalentní

Pole

„*typ*“ **a(n),b(m:n)**
- a(1),…, a(n)
- b(m), … , b(n)

---

podmínky

```
if(„logical“) then
  ...
[else if(„logical“) then
  …]
[else
  ...]
end if
```

```
select case („NE real“)
case („hodnota“)
  …
[case default
  …]
end select
```

---

cyklus

```
do [integer=start,stop,[krok]]
  ...
end do
```

```
do while („logical“)
  …
end do
```

---

cycle – skok na další cyklus
exit – ukončení cyklu

---

náhodná čísla

```
call random_seed()
call random_number(„real“)
```

vstup

```
read(*,*)
read *,
```

---

výstup

```
write(*,*) „něco“
print *, „něco“
```

---

formát

```
write(*,‘(„formát“)‘)
read(*,‘(„formát“)‘)
```

Iw[.m] - integer
Fw.d - real
Ew.d**E**.w – real s exponentem
Lw - logical
n**X** – n mezer
Aw - character

advance=‘no‘ - zustaň na řádku

řetězce

„*integer*“=**len**(„*character*“)
„*integer*“=**len_trim**(„*character*“)
„*character*“=**trim**(„*character*“)
„*character*“ = **char**(„*integer*“)
„*integer*“ = **ichar**(„*character*“)

---

subroutiny

```
subroutine nazev(parametry)
  deklarace proměnných
  příkazy
end subroutine

call nazev(parametry)

return – ukončí subroutinu
```

---

funkce

```
funkce nazev(parametry)
  deklarace proměnných
  příkazy
end function

recursive – udelá z funkce rekurzivní fci
```

result(...) - nastaví výstupní proměnnou

---

standardní funkce

acos(x), **asin**(x), **atan**(x), **atan2**(y,x), **cos**(x), **cosh**(x), **exp**(x), **log**(x) – přirozený log, **log10**(x), **sin**(x), **sinh**(x), **sqrt**(x), **tan**(x), **tanh**(x), **abs**(x), **conjg**(z) – komplexně združené číslo, **dim**(x,y) – kladný rozdíl, **max**(x1, x2, …), **min**(x1, x2, …), **mod**(a,p) – zbytek po dělení se znaménkem jako a, **sign**(x,y) – x se znaménkem y

sum(„pole“) - sečte pole,
product(„pole“) - vynasobí pole,
maxloc(„pole“), **minloc**(„pole“),
maxval(„pole“), **minval**(„pole“)

dot_product(a,b) – skalární součin,
matmul(a,b) – násobení matic,
transpose(a),
reshape(a,“tvar“)

---

soubory

open([unit=]
„*integer*“,**file=**“character“, ...)
- unit je číslo relace

close(„*integer*“) - vstup je číslo relace

inquire(…) - získává informace o souboru či otevřené relaci

---

dynamická alokace

„*typ*“**,allocatable ::**

allocate(„proměnná“(…))

allocated(„proměnná“**)** – test zda již byla proměnná alokována

deallocate(„proměnná“**)**

---

moduly

```
module nezevModulu
  deklarace proměnných
  contains
end module

use nazevModulu
```

---

interface

```
interface
 function nazevFunkce(param)
    deklarace proměnných
  end function
end interface
```

---

volitelné *parametry*

,optional ::
present(„*optional proměnná*“) - test zda byla proměnná na vstupu

---

uživatelský typ

```
type nazevTypu
  definice proměnných
end type

type(nazevTypu) nazevProm

nazev%nazev2
```