Fortran και Αντικειμενοστραφής προγραμματισμός

www.corelab.ntua.gr/courses/fortran_naval/naval

Δδάσκοντες: Άρης Παγουρτζής (pagour@cs.ntua.gr)
(Επίκουρος Καθηγητής ΣΗΜΜΥ)
Δώρα Σούλιου (dsouliou@mail.ntua.gr)
(ΕΔΙΠ ΣΗΜΜΥ)

8η ενότητα

- ✓ Σύνθετοι τύποι δεδομένων
- ✓ Δημιουργία τύπων
- ✓ Στοιχεία αντικειμενοστραφούς προγραμματισμού
- ✓ Fortran 2003

Δημιουργία νέων τύπων δεδομένων (derived)

- `type όνομα_τύπου`
- `    δήλωση 1`
- `    ...`
- `    δήλωση k`
- `end type όνομα_τύπου`
- Όπου **`δήλωση i`** (`i=1,… ,k`) είναι κάποιος γνωστός τύπος δεδομένων (π.χ. **`integer :: a`**). Το όνομα του νέου τύπου δεδομένων ακολουθεί τους γνωστούς κανόνες ονοματολογίας. Π.χ.:

```
type computer_usage_info
    Character (15) :: lastname, firstname
    Integer :: id_number, resource_limit
    Character (6) :: passwd
    Real :: resources_used
end type computer_usage_info
```

Παραδείγματα

- `type Point`
- `    real :: x,y`
- `end type Point`
- Πως ορίζουμε ότι μία μεταβλητή είναι δομή τύπου **Point**;
- `type (Point) :: P,Q`
- Γράφοντας
- `type (computer_usage_info) :: user1,user2`
- ορίζουμε πως οι μεταβλητές **user1** και **user2** είναι δομές τύπου
- **computer_usage_info**
- Ενώ γράφοντας
- `type (computer_usage_info), dimension(50) :: user_list`
  ή
- `type (computer_usage_info) :: user_list(50)`
- ορίζουμε έναν πίνακα 50 θέσεων σε κάθε μία από τις οποίες είναι μία δομή του παραπάνω τύπου

Ανάθεση τιμών σε σύνθετες μεταβλητές

- Οι τιμές που μπορεί να πάρει μια μεταβλητή ενός σύνθετου τύπου δεδομένων είναι ένα σύνολο τιμών για τα επιμέρους πεδία της.
- Για παράδειγμα οι μεταβλητές **P** και **Q** που έχουν οριστεί ως τύπου **Point** μπορούν να πάρουν τιμές ως εξής:
  - **P=Point(2.5,4.7)** ή **P=Point(a,2*b)** όπου **a,b** πραγματικοί
- Για δήλωση σταθεράς π.χ. **center** γράφουμε
  - **Type(Point), Parameter:: center=Point(3.4,8.6)**
- Για πίνακα με στοιχεία τύπου **computer_usage_info** γράφουμε:
  - **User_list(1)= computer_usage_info("Papadopoulos", "Costas", 2345, 700, "passwd1", 240)**

Πρόσβαση στα επιμέρους πεδία

- `structure_name%component_name`

- Για παράδειγμα `P%x` αναφέρεται στη `x`-συντεταγμένη του σημείου.
- Κάθε πεδίο χρησιμοποιείται όπως κάθε άλλη μεταβλητή ίδιου τύπου.

- Απόδοση τιμών σε επιμέρους πεδία:
- `P%x=4.35`

- Είσοδος έξοδος:
- `Read *, user1%lastname`
- `Print *, user2%firstname`

Εμφωλευμένες Δομές

```
Type Point
    real :: x , y
End Type Point

Type Circle
    Type (Point) :: Center
    Real :: radius
End Type Circle

Type(Circle) : C
C=Circle (Point (4.5 , 3.4) , 10.0)
```

- Το C είναι κύκλος με κέντρο το σημείο (4.5 , 3.4) και ακτίνα 10.

Ισοδύναμα:

```
C%Center%x=4.5; C%Center%y=3.4; C%radius=10.0
```

Δημιουργία νέου τύπου δεδομένων: παράδειγμα

προσαρμογή από: [E.J. Akin (2003), Object Oriented Programming via Fortran 90/95]

```
program create_a_type
implicit none
type chemical_element
   character (len=2):: symbol
   integer:: atomic_number
   real:: atomic_mass
end type

type (chemical_element):: argon,carbon,neon
type (chemical_element):: Periodic_Table(109)
real:: mass

carbon%symbol="Ca"
carbon%atomic_number=6
carbon%atomic_mass =12.010
argon=chemical_element("Ar",18,26.98)
```

Δημιουργία νέου τύπου δεδομένων: παράδειγμα

```
write(*,*)"dwse onoma stoixeiou me 2 &
grammata, atomiko arithmo, maza"
read(*,*) neon

Periodic_Table(5)=argon
Periodic_Table(17)=carbon
Periodic_Table(55)=neon

mass=Periodic_Table(5)%atomic_mass

write(*,*) mass
write(*,*) neon
write(*,*) Periodic_Table(17)

end program create_a_type
```

Αν ο χρήστης δώσει:

```
Ne 23 56.87
```

το πρόγραμμα θα τυπώσει:

```
 26.98          !Η ατομική μάζα του argon
Ne 23 56.87     !Τα στοιχεία του neon
Ca 6 12.01      !Τα στοιχεία του carbon
```

Δημιουργία σύνθετου τύπου δεδομένων

προσαρμογή από: [E.J. Akin (2003), Object Oriented Programming via Fortran 90/95]

```
program create_composite_type
implicit none
type chemical_element
   character (len=2)          :: symbol
   integer                    :: atomic_number
   real                       :: atomic_mass
end type
type history
   character (len=31)         :: scientist_name
   integer                    :: year_found
   type (chemical_element)    :: chemelement
end type history
type (chemical_element)       :: argon,carbon,neon,oxygen
type (history)                :: oxygen_history
type (chemical_element)       :: Periodic_Table(109)
real                          :: mass
```

Δημιουργία σύνθετου τύπου δεδομένων

```
argon=chemical_element("Ar",18,26.98)
write(*,*)"dwse onoma (2 letters), atomiko arithmo, maza"
read(*,*) neon
carbon%atomic_mass =12.010; carbon%atomic_number=6
carbon%symbol="Ca"
Periodic_Table(5)=argon
Periodic_Table(17)=carbon
Periodic_Table(55)=neon
mass=Periodic_Table(5)%atomic_mass
write(*,*) mass
write(*,*) neon
write(*,*) Periodic_Table(17)
oxygen=chemical_element("O", 76, 190.2)
oxygen_hist=history ("Joseph Pristley", 1774, oxygen)
write(*,*) oxygen_hist
end program create_composite_type
```

Δημιουργία σύνθετου τύπου δεδομένων

Αν ο χρήστης δώσει: **`Ne 23 56.87`**

Το πρόγραμμα θα τυπώσει:

```
 26.98
 Ne 23 56.87
 Ca 6 12.01
 Joseph Pristley          1774  O   76   190.2
```

Δομές σαν παράμετροι συναρτήσεων

```
PROGRAM POINTS_LINES
Implicit none
Type Point
  real :: x , y
End Type Point
Type(Point) :: P1,P2;  character(1) :: response
```

Δομές σαν παράμετροι συναρτήσεων

```
DO
   write (*,'(1X,A)', Advance="no") &
     "Enter coordinates of points P1,P2: "
   read *, P1%x, P1%y, P2%x, P2%y
   print '(1X,2(A,"(",F5.2,",",F5.2,")"),A,F5.2)',&
     "Distance between ",P1%X,P1%Y," and ",P2%X,P2%Y,&
     " is ", Distance(P1,P2)
   Call FindLine(P1,P2)
   Write (*,*) "More points (y or n)? "
   read *, response
   If (response/="Y") Exit
END DO
Contains
   FUNCTION DISTANCE(P1,P2) ...
   SUBROUTINE FINDLINE(P1,P2) ...
END PROGRAM POINTS_LINES
```

Δομές σαν παράμετροι συναρτήσεων

```
FUNCTION DISTANCE(P1,P2)
  Type(Point), INTENT(IN) :: P1,P2
  Real :: Distance
  Distance=SQRT((P2%X-P1%X)**2+(P2%Y-P1%Y)**2)
END FUNCTION DISTANCE
SUBROUTINE FINDLINE(P1,P2)
  Type(Point), INTENT(IN) :: P1,P2
  Real :: m,b
  IF (P1%x==P2%x)THEN
    PRINT *, "Line is vertical line x= ", P1%X
  ELSE
    m=(P2%Y-P1%Y)/(P2%X-P1%X)
    b=P1%Y-m*P1%X
    PRINT *, "Equation of line is y= ", m, "x+ ", b
  END IF
END SUBROUTINE FINDLINE
```

Ψευδο-OOP στην Fortran 90/95

προσαρμογή από: [Fortran Wiki]

```
module klasi_Rectangle
implicit none
type Rectangle
   real  :: base, height
end type Rectangle
contains
   function rectangle_area(r) result(area)
   type (Rectangle), intent(in):: r
   real :: area
   area=r%base * r%height
   end function rectangle_area
end module klasi_Rectangle
```

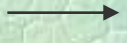
Ψευδο-OOP στην Fortran 90/95

```
module klasi_Circle
implicit none
   real :: pi=3.141592653589931
type Circle
   real  :: radius
end type Circle
contains
   function circle_area(c) result(area)
       type (Circle), intent (in):: c
       real :: area
       area=pi*((c%radius)*(c%radius))
   end function circle_area
end module klasi_Circle
```

Ψευδο-OOP στην Fortran 90/95

προσαρμογή από: [Fortran Wiki]

```
program geometry
use klasi_Circle
use klasi_Rectangle
implicit none
interface compute_area
   module procedure rectangle_area, circle_area
end interface

type (Rectangle)    :: tetragwno
type (Circle)       :: kyklos
real                :: emvado
```



Ψευδο-OOP στην Fortran 90/95

```
. . .
tetragwno=Rectangle(2.1,4.3)
emvado = compute_area(tetragwno)
write(6,100) tetragwno, emvado
100 format("Area of ", f3.1," by ", f3.1," &
   rectangle  is ", f5.2)
kyklos=Circle(5.4)
emvado=compute_area(kyklos)
write(6,200) kyklos, emvado
200 format("Area of circle with ",f3.1, &
   " radius is ",f9.5)
end program geometry
```

ΕΞΟΔΟΣ:

```
Area of 2.1 by 4.3 rectangle  is  9.03
Area of circle with 5.4 radius is  91.60885
```

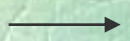
Ψευδο-OOP στην Fortran 90/95 [Fortran Wiki]

```
module class_Circle    !προσαρμογή από [Fortran Wiki]
  implicit none
  private
  public :: Circle, circle_area, circle_print
  real :: pi = 3.14159265358979 31d0 ! Class-wide
                                     ! private constant
  type Circle
     real :: radius
  end type Circle
contains
  function circle_area(this) result(area)
    type(Circle), intent(in) :: this
    real :: area
    area = pi * this%radius**2
  end function circle_area
```

```
 subroutine circle_print(this)
    type(Circle), intent(in) :: this
    real :: area
    area = circle_area(this)
    print *, 'Circle: r = ', this%radius, ' area = ',
   area
  end subroutine circle_print
end module class_Circle

program circle_test
  use class_Circle
  implicit none

  type(Circle) :: c      ! Δήλωση μεταβλητής τύπου
                         !   Circle
  c = Circle(1.4)        ! Χρήση του έμμεσου constructor
                         !   με radius = 1.4
  call circle_print(c)   ! Κλήση υπορουτίνας της κλάσης
end program circle_test
```

OOP στη Fortran 2003

```
module class_Circle !προσαρμογή από [Fortran Wiki]
  implicit none
  private
  real :: pi = 3.1415926535897931d0
       ! Class-wide private constant

  type, public :: Circle
     real :: radius
   contains
     procedure :: area => circle_area
     procedure :: print => circle_print
  end type Circle
```



- Τα αντικείμενα «περιέχουν» και υποπρογράμματα!

OOP στη Fortran 2003

```
contains
  function circle_area(this)  result(area)
    class(Circle), intent(in) :: this
    real :: area
    area = pi * this%radius**2
  end function circle_area

  subroutine circle_print(this)
    class(Circle), intent(in) :: this
    real :: area
    area = this%area()  ! Call type-bound function
    print *, 'Circle: r = ', this%radius, &
       ' area = ', area
  end subroutine circle_print
end module class_Circle
```

OOP στη Fortran 2003

```
program circle_test  !προσαρμογή από [Fortran Wiki]
  use class_Circle
  implicit none
  real :: r
  type(Circle) :: c  ! Δήλωση μετ/τής τύπου Circle

  read *, r          ! Διάβασμα ακτίνας κύκλου
  c = Circle(r)      ! Χρήση του έμμεσου constructor
                     !   με radius = r
  call c%print       ! Κλήση υπορουτίνας του τύπου
end program circle_test
```

OOP στη Fortran 2003: κληρονομικότητα

```
module class_Circle   ! παράδειγμα με χρήση κληρονομικότητας
  implicit none
  private
  real :: pi = 3.14159265358979310
       ! Class-wide private constant

  type, public :: Circle
     real :: radius
   contains
     procedure :: area => circle_area
     procedure :: print => circle_print
  end type Circle
```

OOP στη Fortran 2003: κληρονομικότητα

```
type, public :: Point
   real :: X,Y
end type Point

type, public, extends(Circle) :: Centered_circle
  ! Η κλάση Centered_circle "κληρονομεί" την Circle
  type(Point) :: center
contains
  procedure :: leftmost => centered_circle_leftmost_point
  procedure :: topmost => centered_circle_topmost_point
end type Centered_circle
```

- Χρήση του δηλωτικού **extends**
- Μεταβλητές και υποπρογράμματα της «μητρικής» κλάσης κληρονομούνται *χωρίς να δηλώνονται ξανά*

OOP στη Fortran 2003: κληρονομικότητα

```
contains
  function circle_area(this)  result(area)
    class(Circle), intent(in) :: this
    real :: area
    area = pi * this%radius**2
  end function circle_area

  subroutine circle_print(this)
    class(Circle), intent(in) :: this
    real :: area
    area = this%area()  ! Call type-bound function
    print *, 'Circle: r = ', this%radius, &
        ' area = ', area
  end subroutine circle_print
```

- Αντί για **this** μπορεί να χρησιμοποιηθεί άλλο όνομα.
- Υπενθύμιση ότι η πραγματική παράμετρος είναι το αντικείμενο που περιέχει το υποπρόγραμμα.

OOP στη Fortran 2003: κληρονομικότητα

```
  function centered_circle_leftmost_point(this) result(lpoint)
    class(Centered_circle), intent(in) :: this
    type(Point) :: lpoint
    lpoint%X = this%center%X - this%radius
    lpoint%Y = this%center%Y
  end function

  function centered_circle_topmost_point(this) result(tpoint)
    class(Centered_circle), intent(in) :: this
    type(Point) :: tpoint
    tpoint%X = this%center%X
    tpoint%Y = this%center%Y + this%radius
  end function
end module class_Circle
```

- Υποπρογράμματα με είσοδο αντικείμενο της θυγατρικής κλάσης **`Centered_circle`**

OOP στη Fortran 2003: κληρονομικότητα

```
program circle_test
   use class_Circle
   implicit none
   character:: ans='y'
   real:: r
   type(Point) :: p
   type(Centered_circle) :: c    ! Dhlwsh metavlitis typou
                                 ! Centered_circle
   do while (ans=='y')
     print *,"Give center (X,Y) and radius:"
     read *,p,r
     c = Centered_circle(r,p)    ! constructor with radius r
                                 ! and center p
     print *, "leftmost point: ", c%leftmost()
     print *, "topmost point: ", c%topmost()
     call c%print
     write (*,*) "do you want to repeat (y/n)?"
     read *,ans
   end do
end program circle_test
```

Μετάφραση από ξεχωριστά αρχεία

- Έστω ότι έχουμε γράψει τα modules στο αρχείο **`modfile.f03`** και το κυρίως πρόγραμμα στο αρχείο **`progfile.f03`**. Πώς κάνουμε compilation;

- Δύο τρόποι

  Σε 1 βήμα:

  ```
  gfortran modfile.f03 progfile.f03 -o program
  ```

  Σε 2 βήματα:

  1. **`gfortran -c modfile.f03`**

     παράγεται αρχείο **`modfile.o`** (objective code file)

  2. **`gfortran modfile.o progfile.f03 -o program`**

- Παρόμοια για περισσότερα αρχεία.

Αντικειμενοστραφής προγραμματισμός (i)

- Αντικειμενοστραφές (*object-oriented*) μοντέλο ανάπτυξης λογισμικού
  - Το πρόγραμμα είναι οργανωμένο ως ένα σύνολο από αλληλεπιδρώντα αντικείμενα
  - Κάθε αντικείμενο περιέχει:
    - δεδομένα (*data*), που χαρακτηρίζουν την κατάστασή του
    - μεθόδους (*methods*), δηλαδή κώδικα που υλοποιεί τη συμπεριφορά του
    - ενθυλάκωση (*encapsulation*)

Αντικειμενοστραφής προγραμματισμός (ii)

- Βασικές αρχές
  - Κατά την ανάλυση και τη σχεδίαση, δείτε τα αντικείμενα βάσει της διεπαφής τους (interface) και όχι βάσει της υλοποίησής τους
  - Προσπαθήστε να κρύψετε όσο το δυνατόν μεγαλύτερο μέρος της υλοποίησης
  - Προσπαθήστε να επαναχρησιμοποιήσετε αντικείμενα
  - Προσπαθήστε να ελαχιστοποιήσετε τις διασυνδέσεις μεταξύ αντικειμένων

Αντικειμενοστραφής προγραμματισμός (iii)

- Κύρια χαρακτηριστικά
  - Κληρονομικότητα (inheritance)
  - Πολυμορφισμός (polymorphism)
  - Απόκρυψη πληροφορίας (information hiding)

Αντικειμενοστραφής προγραμματισμός (iv)

- Πλεονεκτήματα
  - Φυσική περιγραφή, τουλάχιστον για ορισμένες περιοχές εφαρμογών
  - Αφαίρεση, δόμηση, επαναχρησιμοποίηση
  - Ιδιαίτερα διαδεδομένος σήμερα, πληθώρα γλωσσών και εργαλείων τον υποστηρίζουν
- Μειονεκτήματα
  - Υπερτιμημένος... ☺
  - Διόγκωση κώδικα