Fortran και Αντικειμενοστραφής προγραμματισμός

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

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

2η ενότητα

- ✓ Δομή προγράμματος
- ✓ Βασικοί τύποι δεδομένων
- ✓ Ανάθεση, είσοδος-έξοδος
- ✓ Τελεστές-προτεραιότητα-παραστάσεις
- ✓ Δομές ελέγχου

Αρχικές Διαφάνειες σε Pascal: Ε. Ζάχος, Ν. Παπασπύρου
Προσαρμογή σε Fortran - συμπληρώσεις: Α. Παγουρτζής, Δ. Σούλιου

Δομή του προγράμματος (i)

```
PROGRAM example
```

Η αρχή και το όνομα του προγράμματος

```
IMPLICIT NONE
REAL:: moires, aktinia
```

χαρακτηριστική δήλωση
δηλώσεις μεταβλητών

```
moires = 180
aktinia = moires * 3.14159 / 180.0
WRITE(*,*)'Η γωνία σε ακτίνια & είναι:',
aktinia
STOP
```

εκτελέσιμες εντολές

```
END
```

τέλος εκτέλεσης

Δομή του προγράμματος (σε C) (i)

```
#include <stdio.h>

int i, j;

void main()
{
    i=15; j=23;
    printf("sum of i and j is: ");
    i=i+j;
    printf("%d", i);
}
```

Δομή του προγράμματος (ii)

PROGRAM <όνομα προγράμματος>

<Κυρίως σώμα>

<Specification Part> (δηλωτικές εντολές)

<Execution Part> (εκτελέσιμες εντολές)

<Subprogram Part> (υπο-προγράμματα)

END ή END PROGRAM <όνομα προγράμματος>

Εντολή **STOP:** τερματίζει την εκτέλεση του προγράμματος. Μπορεί να χρησιμοποιηθεί μία ή περισσότερες φορές. Συχνά εμφανίζεται μόνο πριν το **END** και γι'αυτό παραλείπεται.

Δομή του προγράμματος (iii)

Δηλώσεις μεταβλητών

- μεταβλητή: ένα «κουτί» της μνήμης του υπολογιστή
  όπου μπορεί να αποθηκευτεί
  μια πληροφορία (ένα δεδομένο)
- στο τμήμα δηλώσεων (που βρίσκεται πάντα στην αρχή του προγράμματος) ορίζουμε όλες τις μεταβλητές που χρησιμοποιεί το πρόγραμμα
- για κάθε μεταβλητή ορίζουμε το όνομά της
  και τον τύπο της, δηλαδή το πεδίο των δυνατών τιμών που μπορεί η μεταβλητή να πάρει π.χ.

```
INTEGER:: i,j,k, …
```

Δομή του προγράμματος (iv)

Απλοί τύποι μεταβλητών

- `INTEGER` *ακέραιοι αριθμοί* `0, 1, -3`
- `REAL` *πραγματικοί αριθμοί* `3.14` (τελεία, όχι κόμμα), `2.35E-8` (=2.35×10^{-8})
- `CHARACTER` *χαρακτήρες* `'a', 'C'`
- `CHARACTER (LEN=*)` *string* `'and', "Cool"`
- `LOGICAL` *λογικές τιμές* `.TRUE.` ή `.FALSE.`
- `COMPLEX` *μιγαδικοί αριθμοί* `(-10.0, 8.0)`

Δομή του προγράμματος (v)

Δήλωση μεταβλητών

- `INTEGER:: i, j=12, k, …`
- `REAL:: a, b, x, …`
- `CHARACTER (LEN=4):: mera, minas, etos, …`
- `LOGICAL:: m =.true., n`
- `COMPLEX:: c, e`

Δήλωση σταθερών

- `INTEGER, PARAMETER:: I=10, day=3`
- `REAL, PARAMETER:: PI=3.14159`

Δομή του προγράμματος (vi)

- Σχόλια

```
INTEGER:: x,y  ! Συντεταγμένες κέντρου
REAL:: r       ! Ακτίνα
```

- Κεφαλαίοι ή πεζοί χαρακτήρες;

  Η γλώσσα δεν είναι **case sensitive**

Δομή του προγράμματος (vi)

- Στοίχιση
  - Απαραίτητη για να τρέχει σωστά το πρόγραμμα και να είναι ευανάγνωστο
- Κάθε εντολή σε μία γραμμή.
  - Αν δεν αρκεί μία γραμμή τη συνεχίζω σε περισσότερες προσθέτοντας τον χαρακτήρα & στο τέλος κάθε γραμμής.
  - Αν θέλω περισσότερες από μία εντολές σε μία γραμμή τις χωρίζω με ελληνικό ερωτηματικό ;

Δομή του προγράμματος (vii)

Υποπρογράμματα

```
PROGRAM example

  IMPLICIT NONE
  INTEGER:: i, j, athroisma
  i=15
  j=23
  CALL Prosthesi(i,j,athroisma) !κλήση υποπργ/τος
  WRITE (*,*)'To athroisma einai: ', athroisma
END
```

κυρίως σώμα

```
SUBROUTINE Prosthesi(a,b,athr)
  IMPLICIT NONE
  INTEGER:: a, b, athr
  athr = a+b
END
```

υποπρόγραμμα

Δομή του προγράμματος (σε C) (vii)

Υποπρογράμματα

```
#include <stdio.h>

int i, j;
void add();
{
  i=i+j;
}

void main()
{
  i=15; j=23;
  printf("sum of i and j is: ");
  add();
  printf("%d", i);
}
```

Τί σημαίνει ορθό πρόγραμμα (i)

- Συντακτική ορθότητα
  - το πρόγραμμα πρέπει να υπακούει στους *συντακτικούς* κανόνες της γλώσσας προγραμματισμού
- Συντακτικά σφάλματα στην Fortran
  - εμφανίζονται όταν δεν ικανοποιείται η καθορισμένη σύνταξη
  - παράδειγμα:
  - `(PROGRAM) example`

Τί σημαίνει ορθό πρόγραμμα (ii)

- Νοηματική ορθότητα
  - το πρόγραμμα πρέπει να υπακούει τους *νοηματικούς* κανόνες της γλώσσας προγραμματισμού
- Νοηματικά σφάλματα στην Fortran
  - εσφαλμένη χρήση τελεστών

```
n = 'a' + 1
```

  - χρήση μεταβλητών χωρίς δήλωση

```
IMPLICIT NONE
INTEGER:: n,i
n = i + j
```

αδήλωτη

Τί σημαίνει ορθό πρόγραμμα (iii)

Σημασιολογική ορθότητα

όταν το πρόγραμμα εκτελείται, πρέπει να κάνει ακριβώς αυτό που θέλουμε να κάνει

Σημασιολογικά σφάλματα στην Fortran

προέρχονται από την κακή σχεδίαση ή την κακή υλοποίηση του προγράμματος

αυτά τα σφάλματα ονομάζονται συνήθως bugs και η διαδικασία εξάλειψής τους debugging

```
x = (-b + sqr(b*b-4*a*c))/(2*a)
```

Τί σημαίνει ορθό πρόγραμμα (iv)

Σημασιολογική ορθότητα

όταν το πρόγραμμα εκτελείται, πρέπει να κάνει ακριβώς αυτό που θέλουμε να κάνει

Σημασιολογικά σφάλματα στην Fortran

προέρχονται από την κακή σχεδίαση ή την κακή υλοποίηση του προγράμματος

αυτά τα σφάλματα ονομάζονται συνήθως bugs και η διαδικασία εξάλειψής τους debugging

```
x = (-b + sqr(b*b-4*a*c))/(2*a)
```

sqrt

διαίρεση με το μηδέν

Τί σημαίνει ορθό πρόγραμμα (v)

- Ο μεταγλωττιστής μπορεί να εντοπίσει σε ένα πρόγραμμα την ύπαρξη
  - συντακτικών σφαλμάτων
  - νοηματικών σφαλμάτων
- Τυπώνει κατάλληλα μηνύματα σφάλματος
- Ο προγραμματιστής είναι υπεύθυνος για
  - τη διόρθωση των παραπάνω
  - τον εντοπισμό και τη διόρθωση σημασιολογικών σφαλμάτων

Ανάθεση τιμής σε μεταβλητή

Παραδείγματα αναθέσεων

- `n = 2`
- `pi = 3.14159`
- `done = .true.`
- `ch = 'b'`
- `counter = counter + 1`
- `x1 = (-b + sqrt(b*b-4*a*c))/(2*a)`

Επικοινωνία με το χρήστη (i-a)

Έξοδος στην οθόνη

- `WRITE(*,*)'Hello world'`
- `WRITE(*,*) x`
- `WRITE(*,*) n+1`
- `WRITE(*,*) x, y`
- `WRITE(*,*) 'Η τιμή του x είναι ', x`

Επικοινωνία με το χρήστη (i-b)

Έξοδος με αλλαγή γραμμής

- `WRITE(*,*) 'hello world'`

  !αλλαγή γραμμής μετά το μήνυμα

- `WRITE(*,'(A)',advance='no') 'hello world'`

  !χωρίς αλλαγή γραμμής

- `WRITE(*,'(A/A)') 'hello', 'world '`

  ή, ισοδύναμα

- `WRITE(*,10) 'hello', 'world'`

  `10 FORMAT(A/A)`

  !αλλαγή γραμμής μεταξύ των δύο string

Επικοινωνία με το χρήστη (ii)

Παράδειγμα

```
PROGRAM example
  IMPLICIT NONE
  INTEGER n,m,athroisma
  WRITE(*,*) 'Προσθέτω δυο ακέραιους'
  WRITE(*,*) 'Δώσε το n: '
  READ(*,*) n
  WRITE (*,*) 'Δώσε το m: '
  READ(*,*) m
  athroisma = n + m
  WRITE(*,*)'Το άθροισμα ', n, ' + ', &
     m, ' είναι: ', athroisma
  STOP
END
```

Επικοινωνία με το χρήστη (C) (ii)

Παράδειγμα

```
#include <stdio.h>
void main()
{ int n, m, sum;
  printf("Προσθέτω δύο ακεραίους\n");
  printf("Δώσε το n: ");
  scanf("%d\n",&n);
  printf("Δώσε το m: ");
  scanf("%d\n",&m);
  sum = n + m;
  printf("Το άθροισμα %d + %d είναι:
  ",n,m);
  printf("%d\n",sum);
}
```

Αριθμητικές παραστάσεις (i)

Απλές παραστάσεις

σταθερές και μεταβλητές

Απλές πράξεις

πρόσθεση, αφαίρεση	+, -
πολλαπλασιασμός	*
διαίρεση	/
Ύψωση σε δύναμη	**
πρόσημα	+, -

Αριθμητικές παραστάσεις (ii)

- Προτεραιότητα τελεστών:
  - (α) ** (β) *,/ (γ) +,-
  - π.χ. 5+3*x-y ≡ 5+(3*x)-y
- Προσεταιριστικότητα μεταξύ *,/ και +,-
  - από αριστερά προς τα δεξιά
  - π.χ. x-y+1 ≡ (x-y)+1
- Προσεταιριστικότητα τελεστή **
  - από δεξιά προς αριστερά
  - π.χ. x**y**z ≡ x**(y**z)

Αριθμητικές παραστάσεις (iii)

Σειρά εκτέλεσης των πράξεων

- καθορίζεται εν μέρει από την προτεραιότητα και την προσεταιριστικότητα των τελεστών
- γενικά όμως εξαρτάται και από την υλοποίηση
- *χρησιμοποιείτε παρενθέσεις*: `(x+1)/(y*z)`

Λογικές παραστάσεις (i)

- Συγκρίσεις
  - ισότητα, ανισότητα ==, /=
  - μεγαλύτερο, μικρότερο >, <
  - μεγαλύτερο ή ίσο, μικρότερο ή ίσο >=, <=
- Λογικές πράξεις
  - σύζευξη (και) .AND.
  - διάζευξη (ή) .OR.
  - άρνηση (όχι) .NOT.

Λογικές παραστάσεις (ii)

Πίνακες αλήθειας λογικών πράξεων

p	q	.AND.
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

p	q	p .OR. q
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

p	.NOT. p
FALSE	TRUE
TRUE	FALSE

Λογικές παραστάσεις (iii)

Προτεραιότητα λογικών τελεστών

.NOT. : μεγαλύτερη προτεραιότητα από όλους

.AND. : όπως ο πολλαπλασιασμός

.OR. : όπως η πρόσθεση

π.χ. .NOT. p .AND. q .OR. r
≡ ((.NOT. p) .AND. q) .OR. r

Προσεταιριστικότητα

.NOT. : από δεξιά προς αριστερά

.AND. , .OR. : από αριστερά προς δεξιά

Προτεραιότητα τελεστών συνολικά

1) Αριθμητικοί
2) Σύγκρισης
3) Λογικοί

Παράδειγμα:

```
   x>3+4  .AND.  .NOT.  y==5
≡  x>(3+4)  .AND.  .NOT.  y==5
≡  (x>(3+4))  .AND.  .NOT.  y==5
≡  (x>(3+4))  .AND.  (.NOT.  (y==5))
```

Χρησιμοποιείτε παρενθέσεις!

Δομές ελέγχου

Τροποποιούν τη σειρά εκτέλεσης των εντολών του προγράμματος

Οι εντολές φυσιολογικά εκτελούνται κατά σειρά από την αρχή μέχρι το τέλος

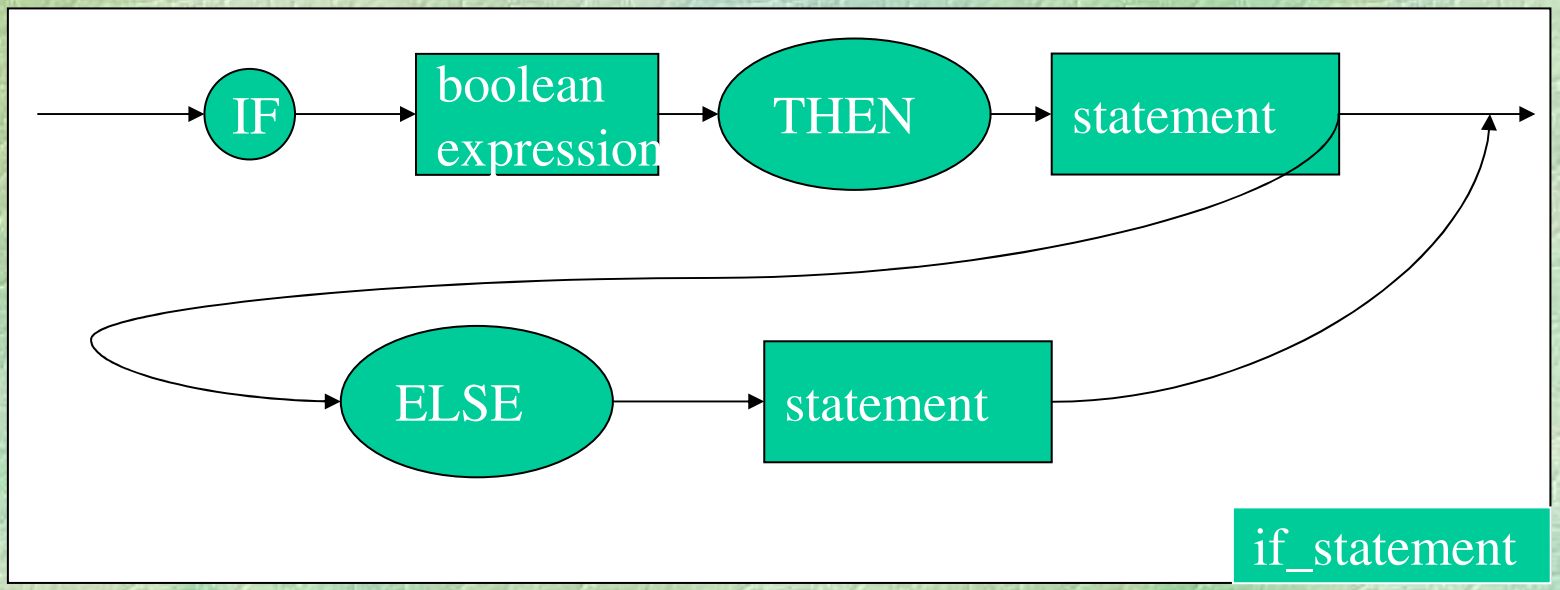
Με τις δομές ελέγχου επιτυγχάνεται:

- εκτέλεση εντολών υπό συνθήκη
- ομαδοποίηση εντολών
- επανάληψη εντολών

Εντολή IF (i)

Εκτέλεση εντολών υπό συνθήκη

Συντακτικό διάγραμμα



Εντολή IF (ii-a)

```
IF (x>10) THEN
  x=x+1
END IF
```

```
IF (age<10) THEN
  WRITE(*,*) 'παιδί'
END IF
```

```
IF ((year>1900).AND.(year<=2000))THEN
  WRITE (*,*) '20ός αιώνας'
END IF
```

```
IF (λογική έκφραση) THEN
     σύνολο εντολών
END IF
```

Η λογική έκφραση αποτιμάται σε .true. ή .false. και μπαίνει σε παρενθέσεις

Εντολή IF (ii-b)

Υπολογισμός δίσεκτου έτους

```
IF    ((MOD(year,4)==0)  .AND. &
       (MOD(year,100)/=0)  .OR.  &
       (MOD(year,400)==0)  .AND. &
       (MOD(year,4000)/=0)) THEN
   WRITE(*,*) 'disekto etos'
END IF
```

Εντολή if (C) (ii)

Παραδείγματα

```
if (x>10) x = x+1;

if (age<10) printf("παιδί");

if ((year>1900) && (year<=2000)
  printf("20ός αιώνας");

if ((year%4==0) &&
   (year%100!=0) ||
   (year%400==0) &&
   (year%4000!=0))
  printf("δίσεκτο έτος");
```

Εντολή IF (iii)

Παραδείγματα (συνέχεια)

```
IF (changed) THEN
   WRITE(*,*) 'Το αρχείο &
      άλλαξε'
   changed = .FALSE.
END IF
```

```
IF (MOD(x,2)==0) THEN
  WRITE(*,*)'άρτιος'
ELSE
  WRITE(*,*)'περιττός'
END IF
```

```
IF (mine) THEN
  me = 1; you = 0
ELSE
  me=0; you=1
END IF
```

```
IF (x > y) THEN
  WRITE(*,*)'μεγαλύτερο'
ELSEIF (x < y) THEN
  WRITE(*,*)'μικρότερο'
ELSE WRITE(*,*) 'ίσο'
END IF
```

Εντολή if (C) (iii)

Παραδείγματα (συνέχεια)

```
if (changed)
{ printf("Το αρχείο άλλαξε\n");
  changed = 0;
}

if (x%2==0) printf("άρτιος");
else printf("περιττός");

if (mine) { me=1; you=0; }
else { me=0; you=1; }

if (x>y) printf("μεγαλύτερο");
else if (x<y) printf("μικρότερο");
else printf("ίσο");
```

Εντολή IF (iv)

Ένα ELSE αντιστοιχεί στο πλησιέστερο προηγούμενο IF που δεν έχει ήδη αντιστοιχιστεί σε άλλο ELSE

Παράδειγμα

```
IF (x>0) THEN
  IF (y>0) THEN
      WRITE(*,*) ' 1ο τεταρτημόριο '
  ELSEIF (y<0) THEN
      WRITE(*,*) '4ο τεταρτημόριο '
  ELSE   ! y=0
      WRITE(*,*) 'άξονας των x '
  END IF
ELSE
  WRITE(*,*) ' αριστερό ημιεπίπεδο '
END IF
```

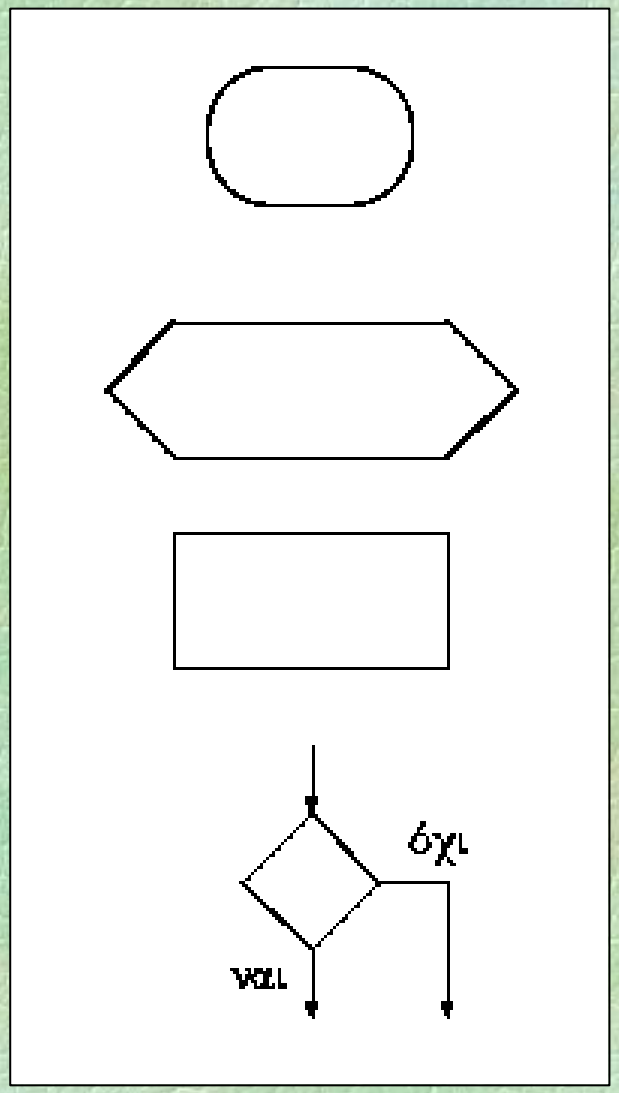
Εντολή if (C) (iv)

Ένα **else** αντιστοιχεί στο πλησιέστερο προηγούμενο **if** που δεν έχει ήδη αντιστοιχιστεί σε άλλο **else**

Παράδειγμα

```
if (x>0)
  if (y>0)
   printf("πρώτο τεταρτημόριο");
  else if (y<0)
   printf("τέταρτο τεταρτημόριο");
  else
   printf("άξονας των x");
else
   printf(" αριστερό ημιεπίπεδο");
```

Λογικά διαγράμματα ροής (i)



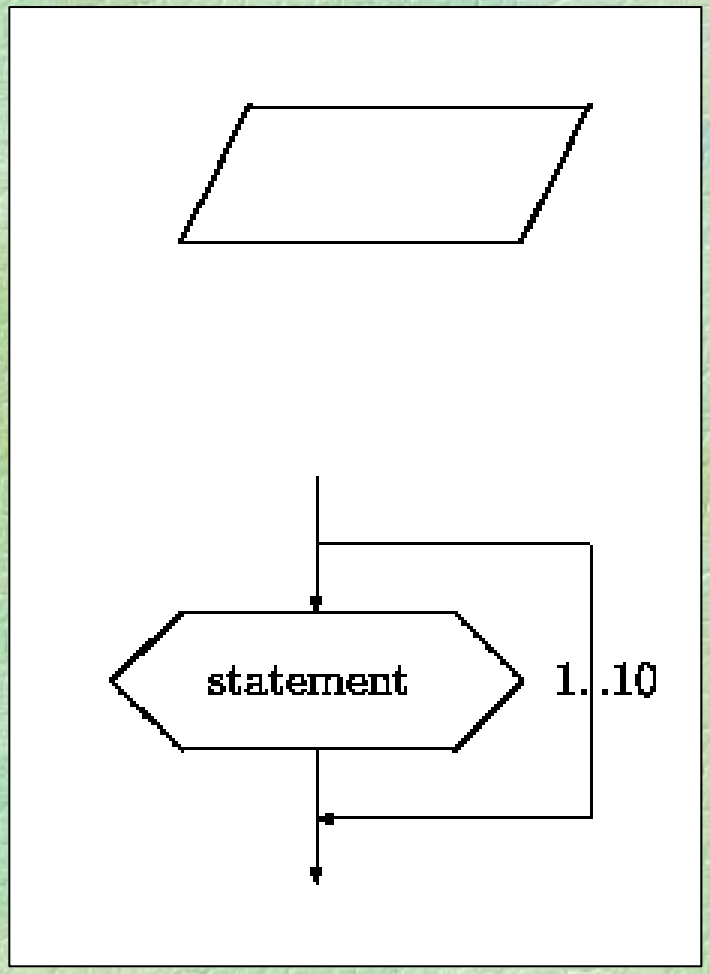
Αρχή και τέλος

Ολόκληρες λειτουργίες ή διαδικασίες

Απλές εντολές

Έλεγχος συνθήκης

Λογικά διαγράμματα ροής (ii)



Λειτουργία εισόδου/εξόδου

Επανάληψη (βρόχος)

Εντολή SELECT CASE (i)

- Εκτέλεση υπό συνθήκη για πολλές διαφορετικές περιπτώσεις
- Προσφέρεται π.χ. αντί του:

```
IF (month==1) THEN
   WRITE(*,*) 'Ιανουάριος'
ELSEIF (month==2) THEN
   WRITE(*,*) 'Φεβρουάριος‘
ELSEIF ...
   ...
ELSEIF (month==12) THEN
   WRITE(*,*) 'Δεκέμβριος '
ELSE
   WRITE(*,*) 'Λάθος Δεδομένα'
END IF
```

Εντολή SELECT CASE (ii)

Το προηγούμενο με SELECT CASE

```
SELECT CASE (month)

  CASE (1)
       WRITE(*,*) 'Ιανουάριος '
  CASE (2)
       WRITE(*,*) 'Φεβρουάριος'
  CASE (3)
       WRITE(*,*) 'Μάρτιος'
  ...
  CASE (12)
       WRITE(*,*) 'Δεκέμβριος '
  CASE DEFAULT
       WRITE(*,*) 'Λάθος Δεδομένα'
END SELECT
```

Εντολή case (C) (ii-a)

Παραδείγματα

```
switch (month)
{
  case  1: printf("Ιανουάριος"); break;
  case  2: printf("Φεβρουάριος"); break;
  case  3: printf("Μάρτιος"); break;
  ...
  case 12: printf("Δεκέμβριος"); break;
}
```

Εντολή case (C) (ii-b)

Παραδείγματα

```
switch (month)
{
  case 1: case 3: case 5: case 7: case 8:
  case 10: case 12:
     printf("31 μέρες"); break;
  case 4: case 6: case 9: case 11:
     printf("30 μέρες"); break;
  case 2:
     printf("28 ή 29"); break;
}
```

Εντολή SELECT CASE (iii)

```
SELECT CASE (selector)          !selector: integer,
  character, logical expression
  CASE (value_list1)            !value_list: εύρος
  τιμών, τιμή ή τιμές
   σύνολο εντολών 1
  CASE (value_list2)
    σύνολο εντολών 2
  CASE (value_list3)
    σύνολο εντολών 3
  ......
  CASE DEFAULT
    σύνολο εντολών n
END SELECT
```

Το case default μπορεί
να μπεί οπουδήποτε
αλλά συνήθως μπαίνει στο τέλος

Παραδείγματα value list case (5), case (3,8), case (5:7), case (:7), case (5:)

Εντολή SELECT CASE (iv)

```
SELECT CASE (class_code)        !selector: integer
  CASE (1)                      !value_list: τιμή
   WRITE (*,*) "jounior"
  CASE (2)
    WRITE (*,*) "senior"
  CASE (3)
    WRITE (*,*) "graduate"
  ......
  CASE DEFAULT
    WRITE (*,*) "Λάθος "
END SELECT
```

Εντολή SELECT CASE (v)

```
SELECT CASE (class_name)                !selector:
  characters
  CASE ("jounior")                      !value_list:
  τιμή
   WRITE (*,*) 1
  CASE ("senior" )
    WRITE (*,*) 2
  CASE ("graduate" )
    WRITE (*,*) 3
  ......
  CASE DEFAULT
    WRITE (*,*) "Λάθος "
END SELECT
```

Εντολή SELECT CASE (vi)

```
SELECT CASE (INT(average))          !selector:
  integer
  CASE (90:)                        !value_list:
  εύρος τιμών
   WRITE (*,*) "grade A"
  CASE (80:89 )
    WRITE (*,*) "grade B"
  CASE (:59)
    WRITE (*,*) "grade C"
  ......

END SELECT
```

Εντολή SELECT CASE (vii)

```
SELECT CASE (index)                    !selector:
  integer
  CASE (high : )                       !value_list:
  εύρος τιμών
      WRITE (*,*) "poor"
  CASE (low : high-1 )
      WRITE (*,*) " fair"
  CASE (: low-1 )
      WRITE (*,*) " good"
END SELECT
```

Εντολή SELECT CASE (viii)

```
SELECT CASE (index)                    !selector:
  integer
  CASE (1,2,3,4)                       !value_list:
  τιμές
      WRITE (*,*) "poor"
  CASE (5,6,7)
      WRITE (*,*) " fair"
  CASE (8,9,10)
      WRITE (*,*) " good"
END SELECT
```

Τι μάθαμε

- Βασικά στοιχεία προγραμματισμού με FORTRAN
  - ✓ Δομή προγράμματος: δηλώσεις, κυρίως σώμα υποπρογράμματα
  - ✓ Βασικοί τύποι δεδομένων: INTEGER, REAL, CHARACTER, LOGICAL, COMPLEX
  - ✓ Ανάθεση: =
  - ✓ Είσοδος-έξοδος: READ, WRITE
  - ✓ Τελεστές: αριθμητικοί, σύγκρισης, λογικοί
  - ✓ Προτεραιότητα-παραστάσεις
  - ✓ Δομές ελέγχου: IF-THEN-ELSE, CASE
- Είμαστε έτοιμοι για τα πρώτα μας προγράμματα!