Πανεπιστήμιο Αιγαίου

Εισαγωγή στην Πληροφορική

Ενότητα 6: Δομές Ελέγχου

Ανδρέας Παπασαλούρος

Τμήμα Μαθηματικών

Σάμος, Μάϊος 2015

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Δομές ελέγχου

- Ακόμη και σε απλά προγράμματα υπάρχει η ανάγκη για την εκτέλεση διαφορετικών εντολών ανάλογα με τις τιμές των δεδομένων του προγράμματος.

Δομές ελέγχου

- Ακόμη και σε απλά προγράμματα υπάρχει η ανάγκη για την εκτέλεση διαφορετικών εντολών ανάλογα με τις τιμές των δεδομένων του προγράμματος.
- Οι τιμές των δεδομένων, ελέγχονται από κατάλληλες *λογικές συνθήκες*.

Δομές ελέγχου

- Ακόμη και σε απλά προγράμματα υπάρχει η ανάγκη για την εκτέλεση διαφορετικών εντολών ανάλογα με τις τιμές των δεδομένων του προγράμματος.
- Οι τιμές των δεδομένων, ελέγχονται από κατάλληλες *λογικές συνθήκες*.
- Διαφορετικές εντολές εκτελούνται ανάλογα με το αν ικανοποιείται ή όχι μια *λογική συνθήκη*.

Δομές ελέγχου

- Ακόμη και σε απλά προγράμματα υπάρχει η ανάγκη για την εκτέλεση διαφορετικών εντολών ανάλογα με τις τιμές των δεδομένων του προγράμματος.
- Οι τιμές των δεδομένων, ελέγχονται από κατάλληλες *λογικές συνθήκες*.
- Διαφορετικές εντολές εκτελούνται ανάλογα με το αν ικανοποιείται ή όχι μια *λογική συνθήκη*.
- Παράδειγμα τέτοιου υπολογισμού αποτελεί η εφαρμογή του ορισμού της απόλυτης τιμής ενός αριθμού:

$$|x| = \begin{cases}$$

Δομές ελέγχου

- Ακόμη και σε απλά προγράμματα υπάρχει η ανάγκη για την εκτέλεση διαφορετικών εντολών ανάλογα με τις τιμές των δεδομένων του προγράμματος.
- Οι τιμές των δεδομένων, ελέγχονται από κατάλληλες *λογικές συνθήκες*.
- Διαφορετικές εντολές εκτελούνται ανάλογα με το αν ικανοποιείται ή όχι μια *λογική συνθήκη*.
- Παράδειγμα τέτοιου υπολογισμού αποτελεί η εφαρμογή του ορισμού της απόλυτης τιμής ενός αριθμού:

$$|x| = \begin{cases} x & \text{αν } x \geq 0 \end{cases}$$

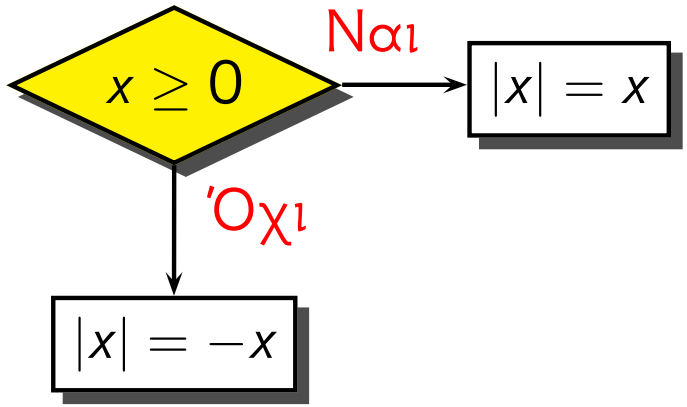
Δομές ελέγχου

- Ακόμη και σε απλά προγράμματα υπάρχει η ανάγκη για την εκτέλεση διαφορετικών εντολών ανάλογα με τις τιμές των δεδομένων του προγράμματος.
- Οι τιμές των δεδομένων, ελέγχονται από κατάλληλες *λογικές συνθήκες*.
- Διαφορετικές εντολές εκτελούνται ανάλογα με το αν ικανοποιείται ή όχι μια *λογική συνθήκη*.
- Παράδειγμα τέτοιου υπολογισμού αποτελεί η εφαρμογή του ορισμού της απόλυτης τιμής ενός αριθμού:

$$|x| = \begin{cases} x & \text{αν } x \geq 0 \\ -x & \text{αλλιώς} \end{cases}$$

Δομές ελέγχου

Διαγραμματικά, ο παραπάνω υπολογισμός είναι δυνατόν να παρασταθεί ως εξής:



Λογικές εκφράσεις

Μια λογική συνθήκη εκφράζεται σε ένα πρόγραμμα Fortran από μια *λογική έκφραση*. Μια *λογική έκφραση* είναι δυνατόν να πάρει μια από τις τιμές `.true.` (αληθής), ή `.false.` (ψευδής).

Απλές λογικές εκφράσεις

Σε μια απλή λογική έκφραση συμμετέχουν οι ακόλουθοι *τελεστές σύγκρισης*:

<	.LT.	μικρότερο
<=	.LE.	μικρότερο ή ίσο
==	.EQ.	ίσο
/=	.NE.	όχι ίσο
>	.GT.	μεγαλύτερο
>=	.GE.	μεγαλύτερο ή ίσο

Παραδείγματα απλών λογικών εκφράσεων

Παραδείγματα λογικών εκφράσεων με τελεστές σύγκρισης είναι:

```
12.5 > 14
a < b
x + y > z - w
characterOption == 'Z'
```

Οι παραπάνω λογικές εκφράσεις αποτιμώνται σε `.true.` ή `.false.`.

Ο τελεστής ισότητας

Προσοχή πρέπει να δοθεί στη διαφορά μεταξύ των τελεστών = και ==. Ο == είναι τελεστής *σύγκρισης* δύο παραστάσεων. Ο = είναι τελεστής *ανάθεσης* της τιμής της παράστασης στα δεξιά του στην μεταβλητή στα αριστερά του τελεστή. Έτσι, η έκφραση

```
2 == 2
```

είναι έγκυρη και αληθής, δηλαδή αποτιμάται στην τιμή `.true.`, ενώ η έκφραση

```
2 = 2
```

δεν είναι έγκυρη, καθώς δεν μπορούμε να αποδώσουμε την τιμή 2 σε μια σταθερά (2).

Σύγκριση χαρακτήρων

Οι τελεστές σύγκρισης εφαρμόζονται τόσο σε αριθμούς όσο και σε χαρακτήρες και συμβολοσειρές. Για παράδειγμα, η έκφραση

```
'A' < 'B'
```

είναι έγκυρη και δηλώνει ότι ο χαρακτήρας `'A'` προηγείται λεξικογραφικά του `'B'`.

Σχέση διάταξης μεταξύ χαρακτήρων

Η σχέση διάταξης μεταξύ χαρακτήρων ορίζεται ως εξής:

1. Το `A` είναι μικρότερο από το `B` ... μικρότερο από το `Z`.
2. Το `0` είναι μικρότερο από το `1` ... μικρότερο από το `9`.
3. Το κενό είναι μικρότερο από τα `A` και το `Z` είναι μικρότερο από το `0`, ή το κενό είναι μικρότερο από το `0` και το `9` είναι μικρότερο από το `A`.
4. Το `9` είναι μικρότερο από το `A`.
5. Το `a` είναι μικρότερο από το `b` ... μικρότερο από το `z`.
6. Το κενό είναι μικρότερο από το `a` και το `z` είναι μικρότερο από το `0`, ή το κενό είναι μικρότερο από το `0` και το `9` είναι μικρότερο από το `a`.

Σχέση διάταξης μεταξύ χαρακτήρων

Με βάση την παραπάνω διάταξη χαρακτήρων ορίζεται σχέση διάταξης μεταξύ συμβολοσειρών (λεξικογραφική διάταξη).
Έτσι, η έκφραση `'Anna' < 'Zoe'` είναι αληθής (`.true.` και δηλώνει ότι η συμβολοσειρά `'Anna'` προηγείται λεξικογραφικά της `'Zoe'`.

Λογικοί τελεστές

Σύνθετες λογικές εκφράσεις της παραπάνω μορφής είναι δυνατόν να προκύψουν με βάση τα παραπάνω είδη απλών εκφράσεων με χρήση *λογικών τελεστών*. Οι κυριότεροι λογικοί τελεστές είναι οι ακόλουθοι:

`.and.`	Λογική σύζευξη
`.or.`	Λογική διάζευξη
`.not.`	Λογική άρνηση

Οι τελεστές `.and.` και `.not.` έχουν τη γνωστή σημασία της λογικής σύζευξης και διάζευξης. Ο τελεστής `.not.` εφαρμόζεται σε μια λογική έκφραση και δίνει τη λογική άρνησή της. Ως παράδειγμα, η έκφραση `1 > 3` είναι ψευδής, άρα η η έκφραση `.not. 1 > 3` είναι αληθής.

Παραδείγματα σύνθετων εκφράσεων

Έκφραση	Τιμή
`1 < 0 .and. 3 > 2`	`.false.`
`1 < 0 .and. 2 > 3`	`.false.`
`1 > 0 .or. 2 > 3`	`.true.`
`4 > 1 .and. .not. -1 > 0`	`.true.`

Προτεραιότητα λογικών τελεστών

`.not.`	Υψηλότερη προτεραιότητα
`.and.`	
`.or.`	Χαμηλότερη προτεραιότητα

Προτεραιότητα λογικών τελεστών

Παράδειγμα

Έτσι, η έκφραση

```
a == b .or. b > 7 .and. .not. c >= 5
```

είναι ισοδύναμη με την

```
a == b .or. (b > 7 .and. (.not. c >= 5))
```

Η απλή εντολή if

Με την εντολή if υποστηρίζεται η υπό συνθήκη εκτέλεση εντολών από ένα πρόγραμμα. Στην απλούστερη μορφή της η if έχει την παρακάτω σύνταξη:

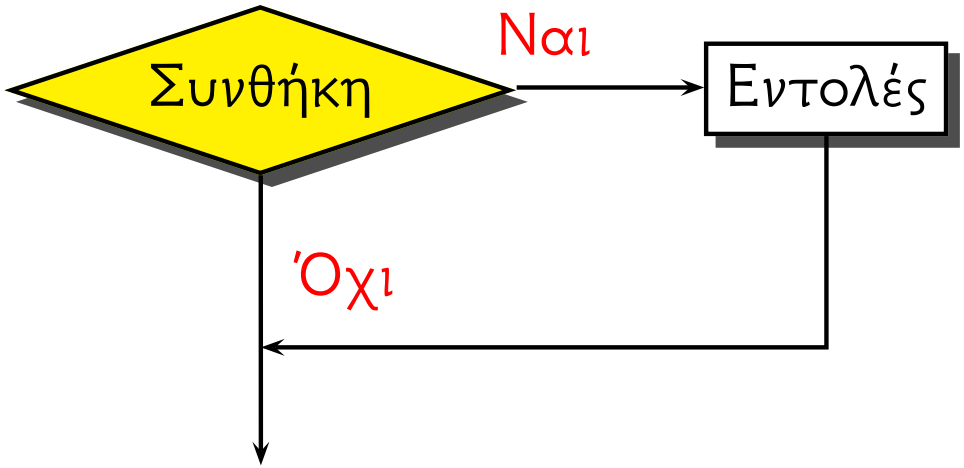
```
IF (λογική_έκφραση) THEN
εντολές...
END IF
```

Στο παραπάνω οι εντολές εκτελούνται μόνο αν η λογική έκφραση είναι αληθής. Αν η έκφραση είναι ψευδής, οι εντολές δεν εκτελούνται και το πρόγραμμα συνεχίζει την εκτέλεσή του στην επόμενη εντολή.

Η απλή εντολή if
Παράδειγμα

```
PROGRAM if_test

INTEGER value

PRINT*, "Δώσε έναν αριθμό:"
READ*, value

IF (value > 0) THEN
  PRINT*, "Ο αριθμός είναι θετικός."
END IF

PRINT*, 2 * value

END PROGRAM if_test
```

Η απλή εντολή `if` μπορεί να παρασταθεί σε μορφή διαγράμματος ως εξής:



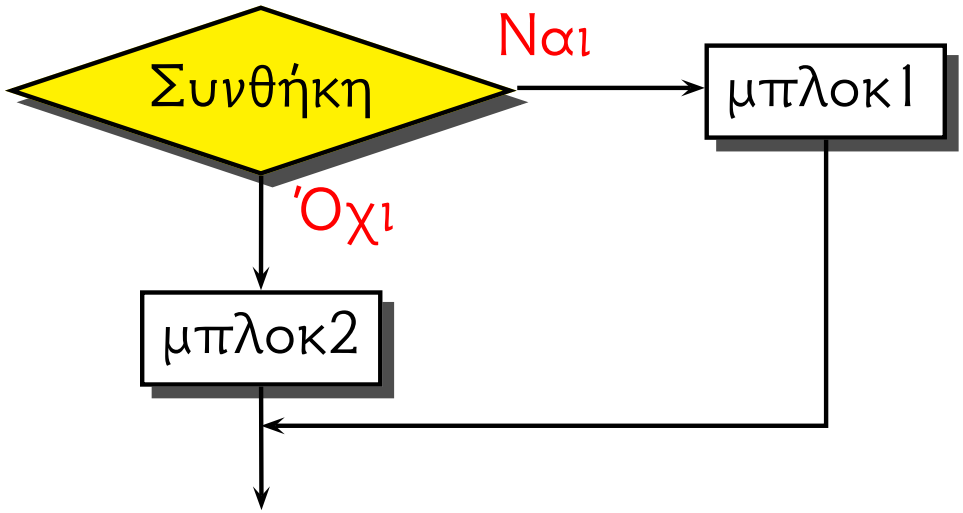
Η εντολή if-else

Η εντολή if-else

```
IF (συνθήκη) THEN
μπλοκ1
ELSE
μπλοκ2
END IF
```

Η εντολή if-else

```
IF (συνθήκη) THEN
μπλοκ1
ELSE
μπλοκ2
END IF
```



Στο παραπάνω εάν η *συνθήκη* ικανοποιείται, τότε εκτελούνται οι εντολές του *μπλοκ1*, αλλιώς εκτελούνται οι εντολές του *μπλοκ2*.

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute



end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none




end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none


read*, value




end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none

print*, "Δώσε έναν αριθμό:"
read*, value




end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none

print*, "Δώσε έναν αριθμό:"
read*, value
if (value >= 0) then



end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none

print*, "Δώσε έναν αριθμό:"
read*, value
if (value >= 0) then
  abs = value




end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none

print*, "Δώσε έναν αριθμό:"
read*, value
if (value >= 0) then
  abs = value
else



end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none

print*, "Δώσε έναν αριθμό:"
read*, value
if (value >= 0) then
  abs = value
else
  abs = -value



end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none

print*, "Δώσε έναν αριθμό:"
read*, value
if (value >= 0) then
  abs = value
else
  abs = -value
end if


end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none

print*, "Δώσε έναν αριθμό:"
read*, value
if (value >= 0) then
  abs = value
else
  abs = -value
end if
print*, "Η απόλυτη τιμή του αριθμού είναι: "
print*, abs
end program absolute
```

Η δομή if-else

Υπολογισμός απόλυτης τιμής αριθμού

```
program absolute
implicit none
integer :: value, abs
print*, "Δώσε έναν αριθμό:"
read*, value
if (value >= 0) then
  abs = value
else
  abs = -value
end if
print*, "Η απόλυτη τιμή του αριθμού είναι: "
print*, abs
end program absolute
```

Η δομή if-else

Εύρεση του μεγαλύτερου μεταξύ δύο αριθμών

Παράδειγμα

Να γραφεί πρόγραμμα που θα διαβάζει δύο ακέραιους αριθμούς a, b και στη συνέχεια θα υπολογίζει και θα τυπώνει τον μεγαλύτερο.

Η δομή if-else

Εύρεση του μεγαλύτερου μεταξύ δύο αριθμών

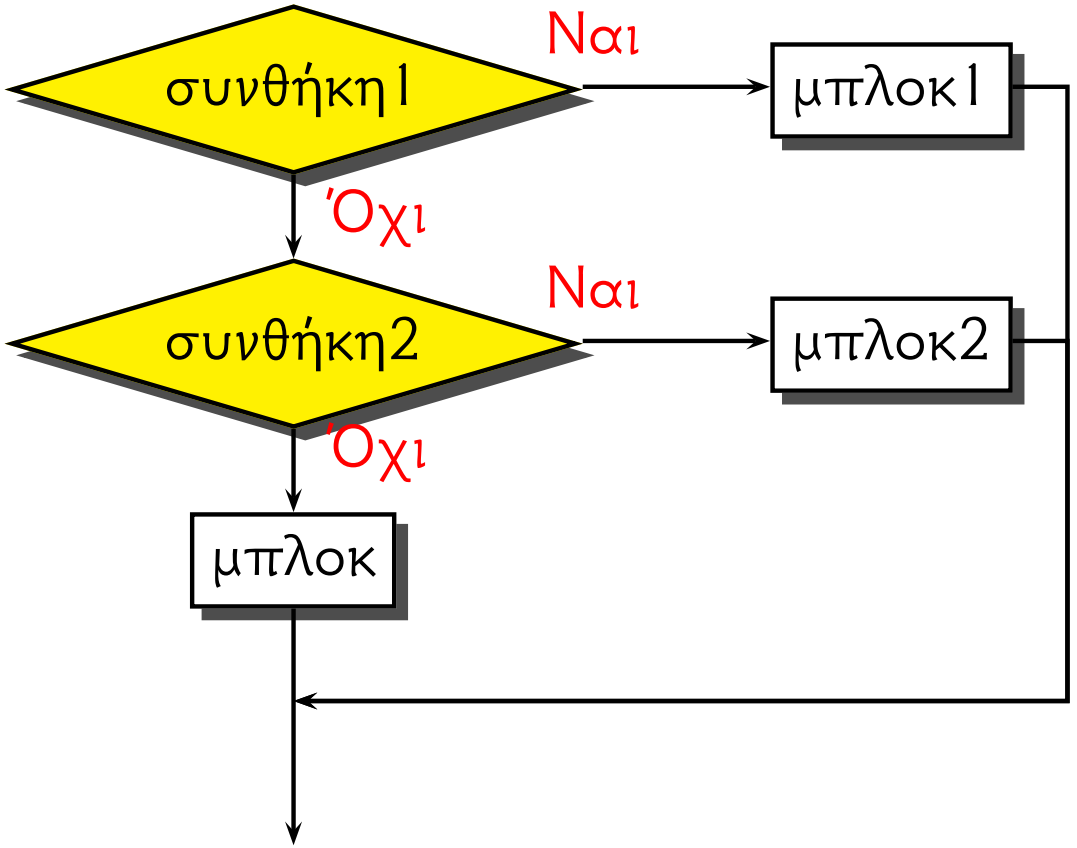
Λύση

```
PROGRAM grater
IMPLICIT NONE
INTEGER :: a, b
PRINT*, 'Δώστε δύο ακέραιους αριθμούς'
READ*, a, b
IF (a > b) THEN
  PRINT*, 'Μεγαλύτερος= ',a
ELSE
  PRINT*, 'Μεγαλύτερος = ',b
END IF
END PROGRAM grater
```

Πολλές φορές απαιτείται η εκτέλεση κατάλληλων εντολών ενώ υπάρχουν περισσότερες από μία επιλογές. Η μορφή αυτή της `if` φαίνεται στη συνέχεια:

```
 IF (συνθήκη1) THEN
μπλοκ1
 ELSE IF (συνθήκη2) THEN
        μπλοκ2
...
 ELSE
μπλόκ
  END IF
```

Στο παραπάνω, εάν επαληθεύεται η *συνθήκη1*, τότε εκτελούνται οι εντολές του *μπλοκ1*, αλλιώς εάν επαληθεύεται η *συνθήκη2*, το *μπλοκ2*, κ.λπ. Εάν καμία από τις συνθήκες δεν επαληθεύεται, τότε εκτελούνται οι εντολές του *μπλοκ*.



Η δομή if-else if
Παράδειγμα

Η δομή if-else if
Παράδειγμα

```
program aprogram

end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none

end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x




end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x
print*,' Διάβασε το x :',x
read*,x



end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x
print*,' Διάβασε το x :',x
read*,x
print*,' Ο :',x, '  είναι:'
```

```
end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x
print*,' Διάβασε το x :',x
read*,x
print*,' Ο :',x, '  είναι:'
if (x>0) then



end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x
print*,' Διάβασε το x :',x
read*,x
print*,' Ο :',x, '  είναι:'
if (x>0) then
  print*,'  θετικός.'



end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x
print*,' Διάβασε το x :',x
read*,x
print*,' Ο :',x, '  είναι:'
if (x>0) then
  print*,'  θετικός.'
else if (x==0) then



end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x
print*,' Διάβασε το x :',x
read*,x
print*,' Ο :',x, '  είναι:'
if (x>0) then
  print*,'  θετικός.'
else if (x==0) then
  print*,' μηδέν.'



end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x
print*,' Διάβασε το x :',x
read*,x
print*,' Ο :',x, '  είναι:'
if (x>0) then
  print*,'  θετικός.'
else if (x==0) then
  print*,' μηδέν.'
else


end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x
print*,' Διάβασε το x :',x
read*,x
print*,' Ο :',x, '  είναι:'
if (x>0) then
  print*,'  θετικός.'
else if (x==0) then
  print*,' μηδέν.'
else
  print*,' αρνητικός'

end program aprogram
```

Η δομή if-else if
Παράδειγμα

```
program aprogram
implicit none
integer :: x
print*,' Διάβασε το x :',x
read*,x
print*,' Ο :',x, '  είναι:'
if (x>0) then
  print*,'  θετικός.'
else if (x==0) then
  print*,' μηδέν.'
else
  print*,' αρνητικός'
end if
end program aprogram
```

Εμφωλευμένες εντολές if

Επιτρέπεται η χρήση μιας εντολής if στο εσωτερικό μιας άλλης εντολής if. Αυτή η δομή ονομάζεται *εμφώλευση* ή *φώλιασμα*.
Στην περίπτωση εμφώλευσης, προτείνεται η χρήση εσοχών (indentation) για την παρακολούθηση της δομής των φωλιασμένων if.

Εμφωλευμένες εντολές if

Παράδειγμα

```
IF (i < 0) THEN
  IF (j < 0) THEN
    x = 0.0
    y = 0.0
  ELSE
    z = 0.0
  END IF
ELSE IF (k < 0) THEN
  z = 1.0
ELSE
  x = 1.0
  y = 1.0
END IF
```

Εμφωλευμένες εντολές if

Εύρεση ρίζας διωνύμου

Να γραφεί πρόγραμμα το οποίο διαβάζει τους συντελεστές α και β της εξίσωσης $ax + b = 0$ και τυπώνει τη λύση της εξίσωσης, αν αυτή υπάρχει ή κατάλληλο μήνυμα.

Λύση

```
program protovathmia


end program protovathmia
```

Λύση

```
program protovathmia
implicit none



end program protovathmia
```

Λύση

```
program protovathmia
implicit none

print*, "a = "; read*, a
print*, "b = "; read*, b

end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
```

```
end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
if (a==0) then



end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
if (a==0) then
  if (b==0) then




end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
if (a==0) then
  if (b==0) then
    print*, ' Η εξίσωση είναι αόριστη'
```

```
end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
if (a==0) then
  if (b==0) then
    print*, ' Η εξίσωση είναι αόριστη'
  else



end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
if (a==0) then
  if (b==0) then
    print*, ' Η εξίσωση είναι αόριστη'
  else
    print*,' Η εξίσωση είναι αδύνατη'



end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
if (a==0) then
  if (b==0) then
    print*, ' Η εξίσωση είναι αόριστη'
  else
    print*,' Η εξίσωση είναι αδύνατη'
  end if



end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
if (a==0) then
  if (b==0) then
    print*, ' Η εξίσωση είναι αόριστη'
  else
    print*,' Η εξίσωση είναι αδύνατη'
  end if
else


end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
if (a==0) then
  if (b==0) then
    print*, ' Η εξίσωση είναι αόριστη'
  else
    print*,' Η εξίσωση είναι αδύνατη'
  end if
else
  print*,'x=',-b/a

end program protovathmia
```

Λύση

```
program protovathmia
implicit none
real :: a,b
print*, "a = "; read*, a
print*, "b = "; read*, b
if (a==0) then
  if (b==0) then
    print*, ' Η εξίσωση είναι αόριστη'
  else
    print*,' Η εξίσωση είναι αδύνατη'
  end if
else
  print*,'x=',-b/a
end if
end program protovathmia
```

Η δομή case

Η Fortran παρέχει μια ακόμη δομή για την επιλογή ανάμεσα από πολλές επιλογές, η οποία μοιάζει με την δομή `if`. Η κύρια διαφορά μεταξύ των δομών `if` και `case` είναι ότι για την δεύτερη μόνο μια έκφραση αποτιμάται ώστε να ελεγχθεί, η οποία παίρνει μια από ένα σύνολο προκαθορισμένων τιμών. Η μορφή της δομής σασε είναι η ακόλουθη:

```
SELECT CASE (έκφραση)
  CASE επιλογή1
  μπλοκ1
  CASE επιλογή2
  μπλοκ2
  ...
  END SELECT
```

Στην παραπάνω μορφή, η *έκφραση* μπορεί να είναι ακέραιου, χαρακτήρα ή λογικού τύπου. Κάθε *επιλογή* πρέπει να είναι του ίδιου τύπου με την έκφραση αλλιώς *προκύπτει* συντακτικό σφάλμα.

Στην παραπάνω μορφή, η *έκφραση* μπορεί να είναι ακέραιου, χαρακτήρα ή λογικού τύπου. Κάθε *επιλογή* πρέπει να είναι του ίδιου τύπου με την έκφραση αλλιώς *προκύπτει* συντακτικό σφάλμα.
Στην απλούστερη μορφή της, η *επιλογή* είναι μια σταθερά, ακέραια, λογική ή χαρακτήρα, η οποία περιέχεται σε παρενθέσεις.

Για παράδειγμα

```
INTEGER:: x = 1

SELECT CASE (x)
  CASE (1)
    PRINT*, "one"
  CASE (2)
    PRINT*, "two"
  CASE (3)
    PRINT*, "three"
END SELECT
```

Για χαρακτήρα ή ακέραιο τύπο η έκφραση μπορεί να είναι διάστημα τιμών ως εξής: `case (low:high)` όπου `low` το κάτω όριο του διαστήματος και `high` το πάνω όριο. Είτε η `low` είτε η `high` είναι δυνατόν να παραληφθούν, αλλά όχι και οι δύο. Αν λείπει η `high` τότε η αντίστοιχη `case` επιλέγεται όταν η *έκφραση* παίρνει τιμή μεγαλύτερη από ή ίση με την τιμή `low`, ενώ αν λείπει η `low` τότε η αντίστοιχη `case` επιλέγεται όταν η *έκφραση* είναι μικρότερη από ή ίση με την τιμή `high`.

Δείτε το επόμενο παράδειγμα:

```
SELECT (NUMBER)
  CASE (:-1) ! Όλες οι τιμές μικρότερες ή ίσες με το -1
    prosimo = -1
  CASE (0) ! Μόνο η τιμή 0
    prosimo = 0
  CASE (1:) ! Όλες οι τιμές μεγαλύτερες ή ίσες με το 1
    prosimo = 1
END SELECT
```

Η γενική μορφή μιας επιλογής (δίπλα στη λέξη case) είναι μια λίστα από συγκεκριμένες τιμές και διαστήματα τιμών που δεν επικαλύπτονται και τα οποία χωρίζονται με κόμμα (,) και περικλείεται από παρενθέσεις, όπως

```
CASE (1, 2, 4:5, 9:)
```

Η μορφή case default αντιστοιχεί στις τιμές που δεν περιλαμβάνονται στις υπόλοιπες επιλογές μιας δομής case, όπως φαίνεται στο παρακάτω παράδειγμα:

```
SELECT (ch)
  CASE ('c', 'd', 'r')
    ch_type = .true.
  CASE ('i':'n')
    int_type = .true.
  CASE DEFAULT !Ο χαρακτήρας δεν ισούται με τίποτε από τα παραπάνω
    real_type = .true.
END SELECT
```

Παράδειγμα

Ένα έτος είναι δίσεκτο όταν ένα από τα παρακάτω είναι αληθή:

Παράδειγμα

Ένα έτος είναι δίσεκτο όταν ένα από τα παρακάτω είναι αληθή:

- Διαιρείται με το 4 αλλά όχι με το 100.

Παράδειγμα

Ένα έτος είναι δίσεκτο όταν ένα από τα παρακάτω είναι αληθή:

- Διαιρείται με το 4 αλλά όχι με το 100.
- Διαιρείται με το 400 αλλά όχι με το 4000.

```
diair_100 = mod(etos,100) == 0
```

```
diair_100 = mod(etos,100) == 0
diair_400 = mod(etos,400) == 0
```

```
diair_100 = mod(etos,100) == 0
diair_400 = mod(etos,400) == 0
diair_4000 = mod(etos, 4000) ==0

                                    ) then
```

```
diair_100 = mod(etos,100) == 0
diair_400 = mod(etos,400) == 0
diair_4000 = mod(etos, 4000) ==0
if (
                                        ) then
   print*,'Δίσεκτο'
else
   print*,''Οχι δίσεκτο'
endif
```

```
diair_100 = mod(etos,100) == 0
diair_400 = mod(etos,400) == 0
diair_4000 = mod(etos, 4000) ==0
if (diair_4 .and. .not. diair_100 .or. &
    diair_400 .and. .not. diair_4000) then
   print*,'Δίσεκτο'
else
   print*,''Οχι δίσεκτο'
endif
```

```
program disekto

diair_100 = mod(etos,100) == 0
diair_400 = mod(etos,400) == 0
diair_4000 = mod(etos, 4000) ==0
if (diair_4 .and. .not. diair_100 .or. &
    diair_400 .and. .not. diair_4000) then
   print*,'Δίσεκτο'
else
   print*,''Οχι δίσεκτο'
endif
end program disekto
```

```
program disekto
implicit none



diair_100 = mod(etos,100) == 0
diair_400 = mod(etos,400) == 0
diair_4000 = mod(etos, 4000) ==0
if (diair_4 .and. .not. diair_100 .or. &
    diair_400 .and. .not. diair_4000) then
   print*,'Δίσεκτο'
else
   print*,''Οχι δίσεκτο'
endif
end program disekto
```

```
program disekto
implicit none


print*, 'Δώστε το έτος:'; read*,etos
diair_4 = mod(etos,4) == 0
diair_100 = mod(etos,100) == 0
diair_400 = mod(etos,400) == 0
diair_4000 = mod(etos, 4000) ==0
if (diair_4 .and. .not. diair_100 .or. &
    diair_400 .and. .not. diair_4000) then
   print*,'Δίσεκτο'
else
   print*,''Οχι δίσεκτο'
endif
end program disekto
```

```
program disekto
implicit none
integer :: etos

print*, 'Δώστε το έτος:'; read*,etos
diair_4 = mod(etos,4) == 0
diair_100 = mod(etos,100) == 0
diair_400 = mod(etos,400) == 0
diair_4000 = mod(etos, 4000) ==0
if (diair_4 .and. .not. diair_100 .or. &
    diair_400 .and. .not. diair_4000) then
   print*,'Δίσεκτο'
else
   print*,''Οχι δίσεκτο'
endif
end program disekto
```

```
program disekto
implicit none
integer :: etos
logical :: diair_4, diair_100, diair_400, diair_4000
print*, 'Δώστε το έτος:'; read*,etos
diair_4 = mod(etos,4) == 0
diair_100 = mod(etos,100) == 0
diair_400 = mod(etos,400) == 0
diair_4000 = mod(etos, 4000) ==0
if (diair_4 .and. .not. diair_100 .or. &
    diair_400 .and. .not. diair_4000) then
   print*,'Δίσεκτο'
else
   print*,''Οχι δίσεκτο'
endif
end program disekto
```