Πανεπιστήμιο Αιγαίου

Εισαγωγή στην Πληροφορική

Ενότητα 7: Δομές Επανάληψης

Ανδρέας Παπασαλούρος

Τμήμα Μαθηματικών

Σάμος, Μάϊος 2015

CC BY NC ND

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

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

Τα προγράμματα των υπολογιστών επεξεργάζονται συχνά δεδομένα με επαναληπτικό τρόπο. Με άλλα λόγια, μια ακολουθία (μπλοκ) εντολών είναι δυνατόν να εκτελείται περισσότερες από μια φορές. Η Fortran διαθέτει την δομή do για την επαναληπτική εκτέλεση εντολών. Η χρήση της δομής αυτής παρουσιάζεται στις επόμενες ενότητες.

Η δομή do ορίζει ένα σύνολο εντολών (μπλοκ) το οποίο πρόκειται να εκτελεστεί επαναληπτικά μέσα σε ένα πρόγραμμα.
Ένα μπλοκ επανάληψης σχηματίζει έναν *βρόχο*.
Ένα παράδειγμα της χρήσης της do είναι το ακόλουθο:

```
DO i = 1, 5
  PRINT*, i
END DO
```

Στο παράδειγμα, μεταξύ των εντολών do και end do περιέχεται μία εντολή, η print*, i, η οποία τυπώνει την τιμή της μεταβλητής i, ως εξής:

```
 1
 2
 3
 4
 5
```

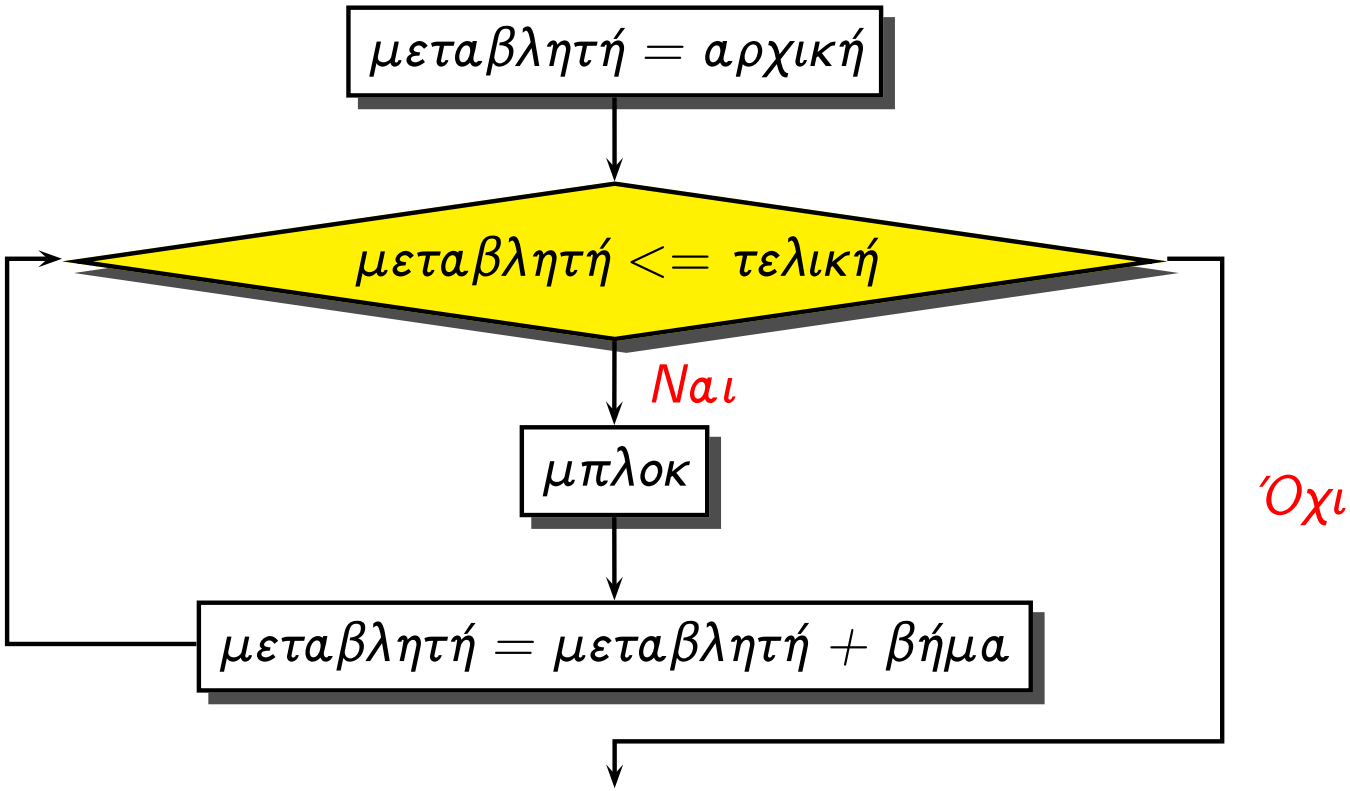
Παρατηρώντας τη συμπεριφορά του παραπάνω προγράμματος, η εντολή `print` εκτελείται 5 φορές. Στην πρώτη επανάληψη, η μεταβλητή `i` έχει την τιμή 1, η οποία σε κάθε επόμενη επανάληψη αυξάνει κατά 1 παίρνοντας τις τιμές $1 \ldots 5$. Οι τιμές της μεταβλητής `i` κατά την εκτέλεση του *βρόχου* ορίζονται στη γραμμή

```
do i = 1, 5
```

όπου δηλώνονται η αρχική (1) και η τελική τιμή (5) για τη μεταβλητή κατά την εκτέλεση της επανάληψης. Υπονοείται ότι σε κάθε επανάληψη, η τιμή της `i` αυξάνει κατά 1.

Στη γενική περίπτωση, η do έχει την ακόλουθη μορφή:

```
DO μεταβλητή = αρχική,τελική [, βήμα]
  μπλοκ
END DO
```



μεταβλητή είναι μια ακέραια μεταβλητή της οποίας η τιμή πρόκειται να αλλάξει σε κάθε επανάληψη. Με αυτή την έννοια, η *μεταβλητή* αποτελεί έναν *μετρητή* της επανάληψης.

αρχική και **τελική** είναι η αρχική και τελική τιμή της μεταβλητής, αντίστοιχα, ενώ *βήμα* είναι μια *προαιρετική* τιμή κατά το την οποία αλλάζει η μεταβλητή του βρόχου σε κάθε επανάληψη, θετική ή αρνητική. Όταν απουσιάζει, η τιμή του βήματος είναι 1.

Το διάγραμμα αναφέρεται στην περίπτωση όπου η αρχική τιμή είναι μικρότερη από ή ίση με την τελική. Στην περίπτωση όπου η τελική τιμή είναι μικρότερη από την αρχική, η λογική έκφραση πρέπει να αντικατασταθεί από την *μεταβλητή* $>=$ *τελική*.

Χαρακτηριστικά παραδείγματα της do παρουσιάζονται στον πίνακα:

Μορφή της do	Αριθμός επαναλήψεων	Τιμές της μεταβλητής
`do i = 2,12,3`		

Χαρακτηριστικά παραδείγματα της do παρουσιάζονται στον πίνακα:

Μορφή της do	Αριθμός επαναλήψεων	Τιμές της μεταβλητής
`do i = 2,12,3`	4	

Χαρακτηριστικά παραδείγματα της do παρουσιάζονται στον πίνακα:

Μορφή της do	Αριθμός επαναλήψεων	Τιμές της μεταβλητής
`do i = 2,12,3`	4	$2, 5, 8, 11$
`do i = 10,1,-2`		

Χαρακτηριστικά παραδείγματα της do παρουσιάζονται στον πίνακα:

Μορφή της do	Αριθμός επαναλήψεων	Τιμές της μεταβλητής
`do i = 2,12,3`	4	$2, 5, 8, 11$
`do i = 10,1,-2`	5	

Χαρακτηριστικά παραδείγματα της do παρουσιάζονται στον πίνακα:

Μορφή της do	Αριθμός επαναλήψεων	Τιμές της μεταβλητής
`do i = 2,12,3`	4	$2, 5, 8, 11$
`do i = 10,1,-2`	5	$10, 8, 6, 4, 2$

Παράδειγμα

Να γραφεί πρόγραμμα Fortran για τον υπολογισμό του αθροίσματος

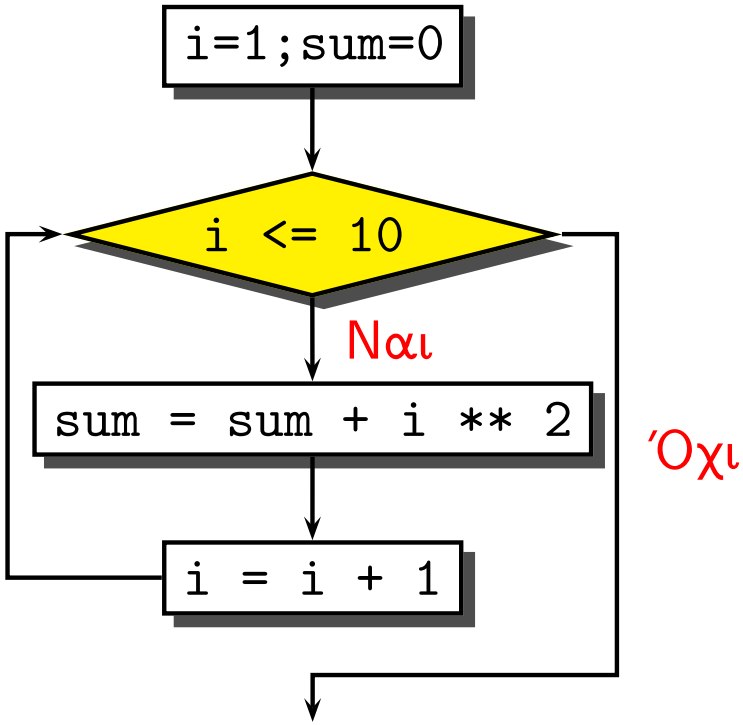
$$\sum_{i=1}^{10} i^2$$

Λύση

Για τον υπολογισμό αθροισμάτων προγραμματιστικά χρησιμοποιείται η εξής τεχνική: Ορίζεται μια μεταβλητή στην οποία αποθηκεύεται το αποτέλεσμα του αθροίσματος: Σε κάθε επανάληψη, η επόμενη νέα τιμή του αθροίσματος ισούται με την παλιά τιμή, συν τον νέο όρο:

```
sum = sum + i ** 2
```

Η σημασία της παραπάνω εντολής είναι: Δώσε ως νέα τιμή στη μεταβλητή `sum` το άθροισμα της παλιάς τιμής της μεταβλητής συν τον όρο `i ** 2`.


i=1;sum=0
i <= 10
Ναι
sum = sum + i ** 2
Όχι
i = i + 1

Η αρχική τιμή της μεταβλητής sum πρέπει να είναι 0, ενώ η μεταβλητή i παίρνει τις τιμές $1, 2, \ldots, 10$. Έτσι, ο κώδικας για τον υπολογισμό του αθροίσματος γίνεται:

```
sum = 0
do i=1,10
  sum = sum + i ** 2
end do
```

Η τελική μορφή του ολοκληρωμένου προγράμματος είναι η ακόλουθη:

```
program athroisma
!  Δήλωση των μεταβλητών i και sum
integer :: i, sum

!  Εκτέλεση της επανάληψης
sum = 0
do i=1,10
  sum = sum + i**2
end do
!  Εκτύπωση του αποτελέσματος
print*, sum
end program athroisma
```

Γενικά, σε ένα βρόχο επανάληψης πρέπει να ορίζονται τα παρακάτω:

- Απόδοση αρχικών τιμών στις μεταβλητές που χρησιμοποιούνται στο βρόχο.
- Συνθήκη τερματισμού του βρόχου.
- Αλλαγή τιμών των μεταβλητών βρόχου.

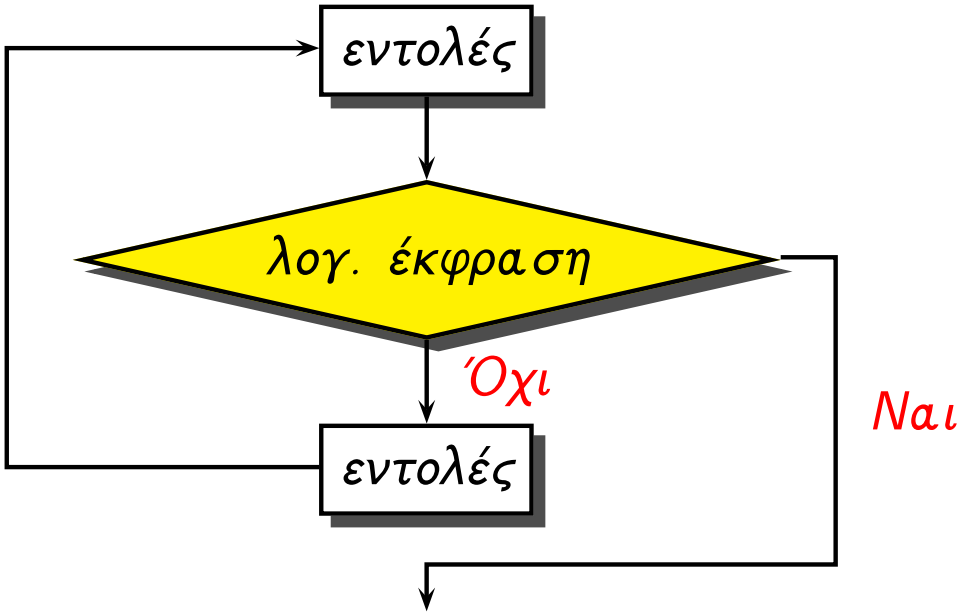
Η γενική μορφή της do

Η παραπάνω μορφή της do προϋποθέτει ότι ο αριθμός των επαναλήψεων είναι εκ των προτέρων γνωστός.

```
DO
  εντολές...
  IF (λογ.  έκφραση) THEN
    EXIT
  END IF
  εντολές...
END DO
```

ή

```
DO
  εντολές...
  IF (λογ.  έκφραση) EXIT
  εντολές...
END DO
```



Στην παραπάνω μορφή είναι προφανές ότι αν μέσα στην do δεν υπάρχει μια εντολή `exit` ή αν η *λογική έκφραση* δεν ικανοποιείται ποτέ, οι εντολές του βρόχου θα εκτελούνται επ΄ άπειρον (ατέρμων βρόχος).

Παράδειγμα

Να γραφεί πρόγραμμα το οποίο διαβάζει μια σειρά από ακεραίους αριθμούς από το πληκτρολόγιο μέχρι να εισαχθεί ο αριθμός -1 και τυπώνει το άθροισμα και τη μέση τιμή τους.

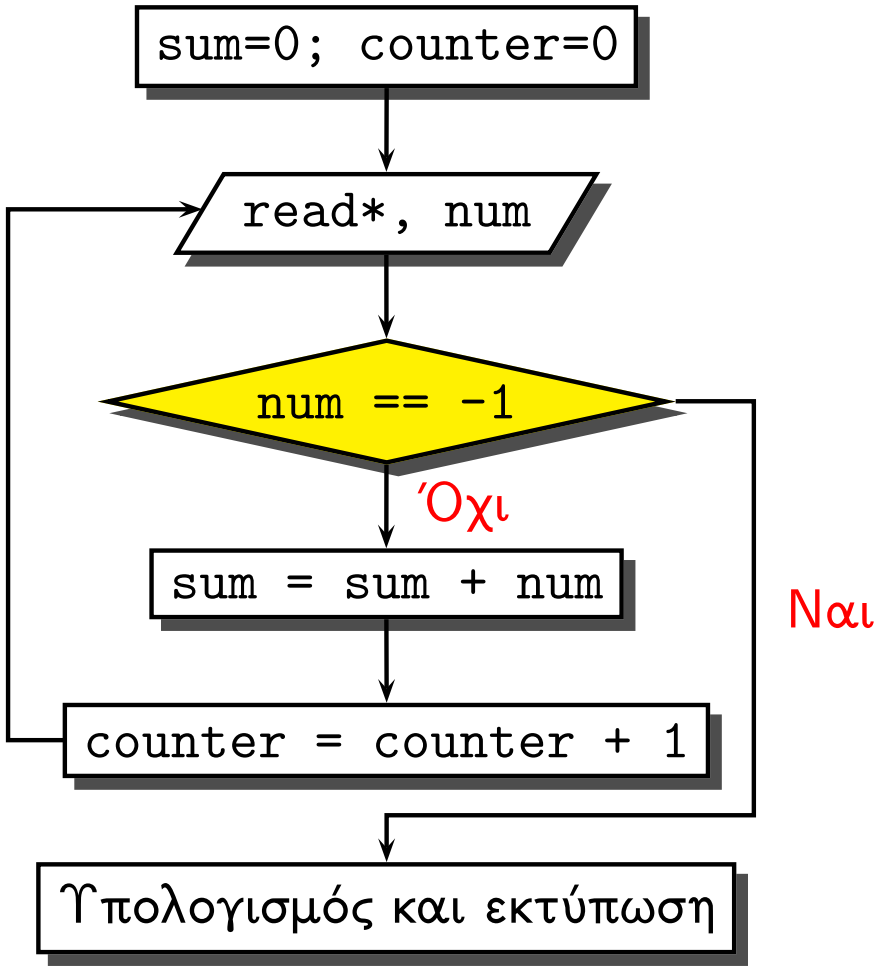
Παράδειγμα

Να γραφεί πρόγραμμα το οποίο διαβάζει μια σειρά από ακεραίους αριθμούς από το πληκτρολόγιο μέχρι να εισαχθεί ο αριθμός -1 και τυπώνει το άθροισμα και τη μέση τιμή τους.

Υπόδειξη

Η εντολή ανάγνωσης των αριθμών τοποθετείται σε μια δομή do. Γίνεται έλεγχος για τη *συνθήκη τερματισμού*: αν ο τρέχων αριθμός που εισάγεται είναι ίσος με το -1. Εάν αυτό αληθεύει, καλείται η εντολή `exit`. Το άθροισμα υπολογίζεται όπως στο παράδειγμα 8 μέσω μιας μεταβλητής `sum`. Για τον υπολογισμό της μέσης τιμής απαιτείται η χρήση μιας ακόμη μεταβλητής η οποία μετράει το πλήθος των αριθμών που εισήχθησαν.

```
PROGRAM NumberInput
IMPLICIT NONE
INTEGER :: num, counter, sum
REAL :: mean
sum = 0; counter = 0
DO
  PRINT*,"Εισάγετε έναν αριθμό (-1 για τερματισμό):  "
  READ*, num
  IF (num == -1) THEN
    EXIT
  END IF
  sum = sum + num
  counter = counter + 1
END DO
mean = REAL(sum) / counter
PRINT*, sum
PRINT*, mean
END PROGRAM NumberInput
```

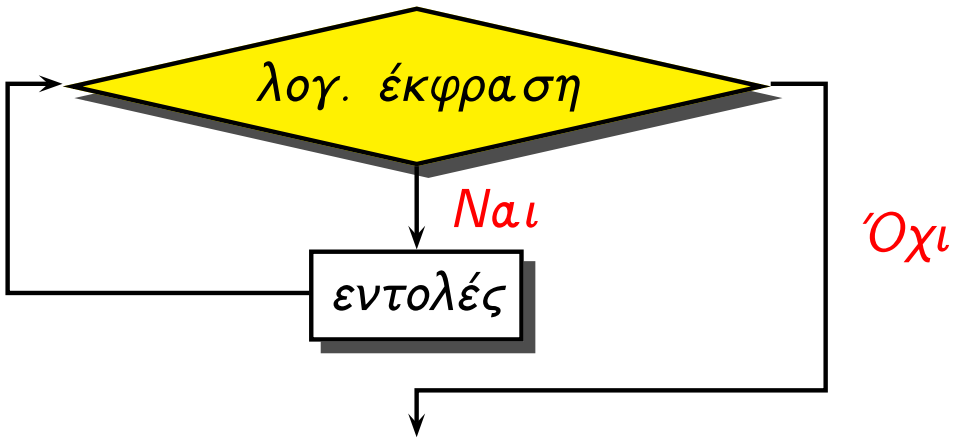

sum=0; counter=0
read*, num
num == -1
Όχι
Ναι
sum = sum + num
counter = counter + 1
Υπολογισμός και εκτύπωση

Η δομή do-while

Η δομή do while εκτελεί ένα μπλοκ εντολών επαναληπτικά όσο μια λογική συνθήκη ικανοποιείται. Η μορφή της εντολής είναι η ακόλουθη:

```
DO WHILE (λογ.  έκφραση)
  εντολές...
END DO
```

Το διάγραμμα που αναπαριστά την παραπάνω μορφή της do είναι το ακόλουθο:



Παράδειγμα

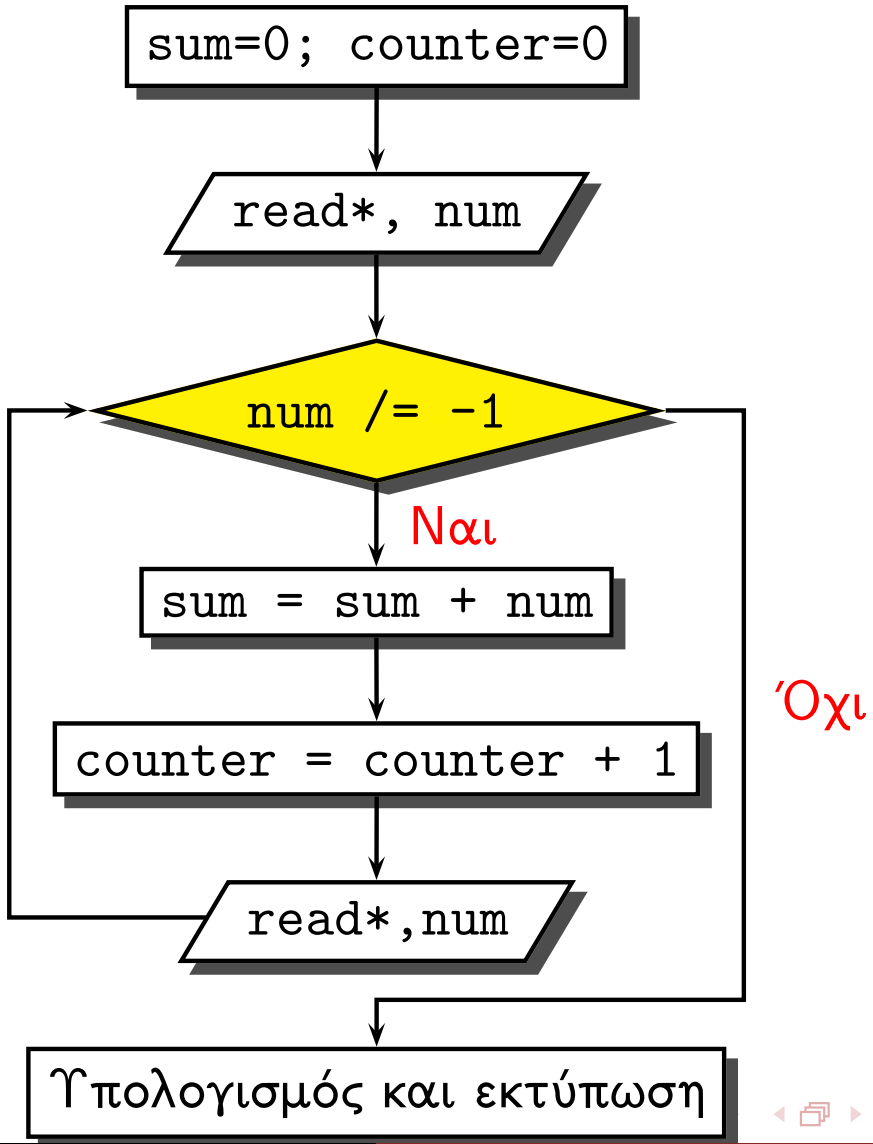
Να γραφεί το πρόγραμμα του παραδείγματος 16 χρησιμοποιώντας τη δομή `do while` αντί της `do`.

```
PROGRAM NumberInput2
  IMPLICIT NONE
  INTEGER :: num, counter, sum
  REAL :: mean

  sum = 0; counter = 0

  PRINT*,"Δώστε έναν αριθμό: "
  READ*, num
  DO WHILE (num /= -1)
    sum = sum + num
    counter = counter + 1
    PRINT*,"Δώστε έναν αριθμό: "
    READ*, num
  END DO
  mean = REAL(sum) / counter
  PRINT*, sum
  PRINT*, mean
END PROGRAM NumberInput2
```

Ένα διάγραμμα με τη ροή εκτέλεσης των εντολών του προγράμματος είναι το ακόλουθο:



Παράδειγμα

Εύρεση του μεγίστου (ή ελαχίστου) μιας ακολουθίας μη αρνητικών αριθμών η οποία διαβάζεται στην είσοδο. Η ακολουθία τερματίζεται όταν εισαχθεί ένας *αρνητικός* αριθμός.

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

```
program maximum



end program maximum
```

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

```
program maximum
implict none



end program maximum
```

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

```
program maximum
implict none



do



end do



end program maximum
```

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

```
program maximum
implict none



do
  print*, "δώσε αριθμό:"; read*, x


end do



end program maximum
```

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

```
program maximum
implict none



do
  print*, "δώσε αριθμό:"; read*, x
  if (x < 0) exit

end do



end program maximum
```

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

```
program maximum
implict none



do
  print*, "δώσε αριθμό:"; read*, x
  if (x < 0) exit
  if (x > max)  max = x
end do



end program maximum
```

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

```
program maximum
implict none



do
  print*, "δώσε αριθμό:"; read*, x
  if (x < 0) exit
  if (x > max)  max = x
end do

print*, "Ο μεγαλύτερος είναι ο ", max

end program maximum
```

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

```
program maximum
implict none


max = -1 ! Ο μικρότερος από τους δυνατούς αριθμούς
do
  print*, "δώσε αριθμό:"; read*, x
  if (x < 0) exit
  if (x > max)  max = x
end do

print*, "Ο μεγαλύτερος είναι ο ", max

end program maximum
```

Εύρεση του μέγιστου μιας ακολουθίας θετικών ακεραίων

```
program maximum
implict none
integer :: x, max

max = -1 ! Ο μικρότερος από τους δυνατούς αριθμούς
do
  print*, "δώσε αριθμό:"; read*, x
  if (x < 0) exit
  if (x > max)  max = x
end do
if (max >=0) then
print*, "Ο μεγαλύτερος είναι ο ", max
else print*,'Δεν δώσατε κανένα θετικό αριθμό'
end program maximum
```

Παράδειγμα

Να γραφτεί πρόγραμμα το οποίο διαβάζει έναν θετικό ακέραιο, n, από το πληκτρολόγιο και υπολογίζει και τυπώνει το $n!$. Αν ο ο αριθμός είναι αρνητικός θα τυπώνεται κατάλληλο μήνυμα

```
program factorial
implicit none
  integer ::  n, i, p
  print*, 'Δώστε το n :'; read*, n
  if (n < 0) then
    print*, ' Το n  πρέπει να είναι θετικό'
    stop
  end if

  p = 1
  ! Αν  n = 0  ή  n = 1  τότε η παρακάτω εντολή   do δεν θα
  !  εκτελεστεί καμία φορά και το p  θα παραμείνει 1.
  do i = 2, n
    p = p * i
  end do
  print*, 'n!= ', p
end program factorial
```

Παράδειγμα

α΄ Να γραφεί πρόγραμμα το οποίο διαβάζει επαναληπτικά ακέραιους από το πληκτρολόγιο, μέχρι να δοθούν συνολικά 3 αρνητικοί αριθμοί.

β΄ Να τροποποιηθεί το παραπάνω πρόγραμμα ώστε να τερματίζει όταν δοθούν δύο διαδοχικοί αρνητικοί αριθμοί. Σε κάθε περίπτωση, το πρόγραμμα θα υπολογίζει και θα τυπώνει τη μέση τιμή των αριθμών.

Παράδειγμα

α′ Να γραφεί πρόγραμμα το οποίο διαβάζει επαναληπτικά ακέραιους από το πληκτρολόγιο, μέχρι να δοθούν συνολικά 3 αρνητικοί αριθμοί.

β′ Να τροποποιηθεί το παραπάνω πρόγραμμα ώστε να τερματίζει όταν δοθούν δύο διαδοχικοί αρνητικοί αριθμοί. Σε κάθε περίπτωση, το πρόγραμμα θα υπολογίζει και θα τυπώνει τη μέση τιμή των αριθμών.

Υπόδειξη: Για τον έλεγχο των διαδοχικών τιμών, χρησιμοποιήστε μια λογική μεταβλητή η οποία παίρνει την τιμή `.true.` αν κατά την προηγούμενη επανάληψη εισήχθηκε αρνητική τιμή.

```
program threenegs
implicit none

  integer :: arnitikoi,x, sum,counter

  sum = 0; counter = 0
  arnitikoi = 0
  do while (arnitikoi < 3)
    print*,'Give integer: ';   read*, x
    sum=sum + x
    counter=counter + 1
    if (x < 0) then
       arnitikoi = arnitikoi + 1
    end if
  end do
  print*,'mo=',sum/real(counter)
end program threenegs
```

Να τροποποιηθεί το παραπάνω πρόγραμμα ώστε να μην υπολογίζονται οι αρνητικοί στο άθροισμα και το μέσο όρο.

Το πρόγραμμα twoneg

```
program twoneg
implicit none
  integer :: x, sum, counter
  logical :: negprevious
  sum = 0
  counter = 0
  negprevious = .false.
 ! ...
```

Συνέχεια του προγράμματος twoneg

```
!  Συνέχεια ...
  do
    print*,' Δώσε ακέραιο: ';read*, x
    sum = sum + x
    counter = counter + 1
    if ((x < 0).and.(negprevious))   exit
    if (x < 0) then
      negprevious = .true.
    else
      negprevious = .false.
    end if
  end do
  print*, 'Μέσος όρος: ', sum / real(counter)
end program twoneg
```