Ελληνική Δημοκρατία
Τεχνολογικό Εκπαιδευτικό
Ίδρυμα Ηπείρου

Προγραμματισμός Ι

Ενότητα 3 : **Εντολές Επανάληψης**

Αλέξανδρος Τζάλλας

Τμήμα Μηχανικών Πληροφορικής Τ.Ε

Προγραμματισμός Ι

Ενότητα 3 : Εντολές Επανάληψης

Αλέξανδρος Τζάλλας
Λέκτορας
Άρτα, 2015

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
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

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 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.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Ηπείρου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.

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

Σκοποί ενότητας

- Να περιγραφούν αναλυτικά οι δυνατότητες των εντολών επανάληψης.
- Να αναλυθεί με ακρίβεια η χρησιμότητα/σκοπός των εντολών επανάληψης.
- Να περιγραφούν οι συντακτικοί κανόνες των εντολών επανάληψης, **while, repeat** και **for**.
- Να αναλυθεί με ακρίβεια η χρησιμότητα/σκοπός των φωλιασμένων Εντολών Επανάληψης

Περιεχόμενα ενότητας

- Χρησιμότητα/Σκοπός Εντολών Επανάληψης
- Εντολή while
- Εντολή repeat
- Εντολή for
- Φωλιασμένες Εντολές Επανάληψης
- Παραδείγματα

Εντολές Επανάληψης

- Σε ένα πρόγραμμα υπάρχουν περιπτώσεις που απαιτείται η επαναληπτική εκτέλεση μίας ή περισσοτέρων προτάσεων
- Ο αριθμός των επαναλήψεων καθορίζεται από μια συνθήκη τερματισμού
- Η Pascal υποστηρίζει τρείς εντολές επανάληψης:
  - **While**
  - **Repeat**
  - **For**

Η εντολή While$_{1/7}$

- **Σκοπός:** Επαναληπτική εκτέλεση μια πρότασης, όσο ισχύει μια δεδομένη συνθήκη

WHILE Λογική Έκφραση DO Πρόταση

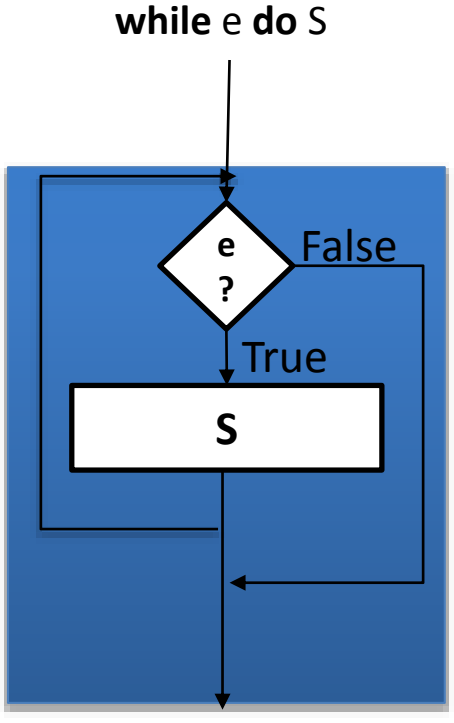
- **Εξηγήσεις:** Αρχικά υπολογίζεται η τιμή της λογικής και αν είναι αληθής (true) εκτελείται η πρόταση
- Η διαδικασία επαναλαμβάνεται όσο (while) η τιμής της λογικής έκφρασης είναι αληθής (true) και σταματά μόλις γίνει ψευδής (false)

```
while abs(x=a/x)> 1E-6 do x:=0.5*(x+a/x);

while c=‘ ’ do read(c);
```

Η εντολή While$_{2/7}$

Η ροή εκτέλεσης της πρότασης:
while e **do** S



Υπόλοιπες προτάσεις προγράμματος

Η εντολή While$_{3/7}$

- **Παρατηρήσεις:** Αν η λογική έκφραση είναι αρχική ψευδής, η πρόταση μετά το do δεν εκτελείται

  Π.χ. η επόμενη πρόταση δεν θα εκτελεστεί καμία φορά :

  x:=-3;

  while x>0 **do** x:=x-1;

- Πρέπει να φροντίζουμε ώστε οι μεταβλητές που καθορίζουν την τιμής λογικής έκφρασης να έχουν την κατάλληλη τιμή πριν την εκτέλεση της εντολής while
- Όταν θέλουμε να διαβάσουμε μια ακολουθία στοιχείων των οποίων δε γνωρίζουμε το πλήθος, χρησιμοποιούμε μια ειδική τιμή που δηλώνει το τέλος της ακολουθίας (ονομάζεται σκοπός: sentinel)

Η εντολή While$_{4/7}$

- **Κανόνες:** Αν απαιτείται η επαναληπτική εκτέλεση μια ακολουθίας προτάσεων τότε σχηματίζουμε μια σύνθετη πρόταση (begin, end)

  Δηλαδή:

  while Λογική Έκφραση **do begin**

  : Ομάδα

  : Προτάσεων

  end;

```
While c<> ‘ ’ begin
              count: =count+1;
              read(c)
              end;
```

- Το πότε θα σταματήσει η επαναληπτική εκτέλεση πρέπει να καθορίζεται μέσα στην πρόταση ή την ακολουθά προτάσεων της εντολής

Η εντολή While 5/7

- **Παράδειγμα 1**

```
program repetition;
var     x: integer;
begin
x:=0;
while x<4 do begin
                    writeln(x);
                    x :=x+1;
                    end
 end.
```

Αποτέλεσμα:
0
1
2
3

Η εντολή While 6/7

- **Παράδειγμα 2:**

Να γραφεί πρόγραμμα που διαβάζει ένα σύνολο αριθμών και να υπολογίζει το άθροισμα των τετραγώνων τους. Το τέλος του συνόλου των αριθμών θα δηλώνεται με κάποιο συγκεκριμένο αριθμό (π.χ. -999)

```
program repetition;
var     x, sum: real;
begin
sum:=0;
readln(x);
while x<>-999 do begin
                    sum:=sum+sqr(x);
                 readln(x);
                 end;
 writeln('square sum =', sum);
 end.
```

Η εντολή While 7/7

- **Παράδειγμα 3:**

Να γραφεί πρόγραμμα που να προσομοιώνει τις λειτουργίες ενός υπολογιστή τσέπης. Δηλαδή να διαβάζει μια ακολουθία από αριθμητικές τιμές και αριθμητικούς τελεστές (+,-,*,/), να υπολογίζει την έκφραση και να τυπώνει το αποτέλεσμα όταν διαβάζει το χαρακτήρα '='

```
program calculator;
var   result, number: real;
      telestis: char;
begin
read(result);
read(telestis);
while telestis <> '=' do
       begin
        read(number);
           case  telestis of
           '+': result:=result+number;
           '-': result:=result-number;
           '*': result:=result*number;
            '/': result:=result/number;
           end;
         read(telestis);
         end;
 writeln(result);
 end.
```

Η εντολή Repeat$_{1/5}$

- **Σκοπός:** Επαναληπτική εκτέλεση μιας ή περισσοτέρων προτάσεων, έως ότου ικανοποιηθεί μια συνθήκη

```
REPEAT
   : Ομάδα
   : Προτάσεων
UNTIL Λογική Έκφραση
```

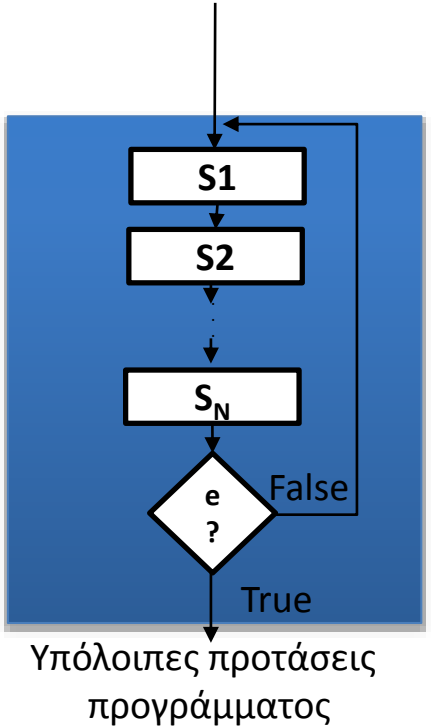
- **Εξηγήσεις:** Η εκτέλεση της εντολής ακολουθεί τα εξής βήματα:

1. Εκτελούνται οι προτάσεις μεταξύ του repeat…until
2. Υπολογίζεται η τιμή της λογικής έκφρασης
3. Αν η τιμή της είναι false επαναλαμβάνεται το βήμα 1 (Αν είναι true σταματά η επαναληπτική εκτέλεση των προτάσεων και η εκτέλεση συνεχίζει με τις υπόλοιπες προτάσεις του προγράμματος

Η εντολή Repeat 2/5

- Η ροή εκτέλεσης της εντολής repeat

(a)
```
repeat
x:=0.5*(x+a/x);
until abs(x-a/x); <=1E-6
```



(b)
```
repeat
count:=count+1;
read(c)
Until c=‘ ’;
```

Η εντολή Repeat 4/5

- **Παράδειγμα 1**

```
program again;
var     x: integer;
begin
x:=0;
        repeat
        writeln(x);
        x:=x+1;
        until x=4;
 write('Telos');
 end.
```

```
program repetition;
var     x: integer;
begin
x:=0;
while x<4 do begin
                writeln(x);
              x:=x+1;
              end
 end.
```

Αποτέλεσμα:
0
1
2
3

Η εντολή Repeat 3/5

- **Παράδειγμα 2:**
  Υπολογισμός του αθροίσματος S=1+3+5+…+N

```
program sumN;
var  s,n,k: real;
begin
write('N='); readln(n);
s:=0;
k:=1;
repeat
s:=s+k;
k:=k+2;
until k>n;
writeln('S=', s);
 end.
```

Η εντολή Repeat 5/5

- **Παράδειγμα 3:**

Να γραφεί το πρόγραμμα προσομοίωσης ενός υπολογιστή τσέπης χρησιμοποιώντας την εντολή repeat

```
program calculator;
var     result, number: real;
        telestis: char;
begin
read(result);
read(telestis);
while telestis <> '=' do
      begin
       read(number);
          case  telestis of
          '+': result:=result+number;
          '-': result:=result-number;
          '*': result:=result*number;
           '/': result:=result/number;
          end;
         read(telestis);
         end;
 writeln(result);
 end.
```

```
program calculator;
var     result, number: real;
        telestis: char;
begin
read(result);
read(telestis);
repeat
          read(number);
           case  telestis of
           '+': result:=result+number;
           '-': result:=result-number;
           '*': result:=result*number;
            '/': result:=result/number;
           end;
         read(telestis);
 until telestis ='=';
 writeln(result);
 end.
```

While vs Repeat 1/3

- Η εντολή while εκτελείται όσο η τιμής της λογικής έκφρασης είναι true ενώ η εντολή repeat εκτελείται όσο η τιμής της λογικής έκφρασης είναι false
- Έτσι αν μια επαναληπτική ακολουθία προτάσεων είναι εκφρασμένη με την εντολή while και θέλουμε να την εκφράσουμε με την εντολή repeat αντιστρέφουμε την λογική έκφραση
- Για παράδειγμα οι ακόλουθες προτάσεις, στις οποίες διαβάζονται θετικοί αριθμοί από την μονάδα εισόδου, υπολογίζεται η τιμής της έκφρασης και τυπώνεται το αποτέλεσμα:

```
read(x);
while x>0 do
begin
y:=2*sqrt(x)+1;
writeln(y);
read(x);
end;
```

```
repeat
read(x);
If x>0 then begin
y:=2*sqrt(x)+1;
writeln(y);
end
until x<=o;
```

While vs Repeat$_{2/3}$

- Η βασική διαφορά των εντολών while και repeat είναι ότι η ακολουθία προτάσεων της εντολής repeat εκτελείται τουλάχιστον μια φορά (έτσι η εντολή repeat ενδείκνυται στις περιπτώσεις όπου απαιτείται μια τουλάχιστον εκτέλεση μιας ακολουθίας προτάσεων για παράδειγμα ένα μενού)
- Η εντολή while υπολογίζει πρώτα την τιμή της λογικής έκφρασης και στη συνέχεια εκτελεί την ακολουθία των προτάσεων (υπάρχουν όμως περιπτώσεις όπου οι τιμές των μεταβλητών που καθορίζουν την τιμής της λογικής έκφρασης είτε είναι απροσδιόριστες ή δεν έχουν τη σωστή τιμή πριν την πρώτης εκτέλεση της ακολουθία προτάσεων

While vs Repeat$_{3/3}$

- Για την υλοποίηση μια τέτοιας περίπτωσης με την εντολή while, θα πρέπει ο προγραμματιστής να θέσει αρχικές τιμές στις μεταβλητές έτσι ώστε η λογική έκφραση να έχει αρχική τιμή

π.χ. **repeat** read(ch) **until** ch='.';

- Αν θέλουμε να εκφράσουμε την πρόταση χρησιμοποιώντας την εντολή while θα έπρεπε να δώσουμε κάποια αρχική τιμή στη μεταβλητή ch

π.χ. ch:='I';

while ch <> '.' do **read(ch)**;

Η εντολή For$_{1/6}$

- **Σκοπός:** Επαναληπτική εκτέλεση μιας πρότασης ή περισσοτέρων προτάσεων για δεδομένο αριθμό επαναλήψεων

FOR Μεταβλητή Ελέγχου :=Αρχική Εκφραση
TO Τελική Εκφραση DO Πρόταση;

FOR Μεταβλητή Ελέγχου :=Αρχική Εκφραση
DOWNTO Τελική Εκφραση DO Πρόταση;

Η εντολή For 2/6

- **Εξηγήσεις:** Μεταβλητή ελέγχου είναι μια οποιαδήποτε μεταβλητή που χρησιμοποιείται σα μετρητής των επαναλήψεων
- Η μεταβλητή ελέγχου μπορεί να είναι οποιουδήποτε τύπου εκτός από **real**
- Αρχική έκφραση είναι μια έκφραση ίδιου τύπου με τη μεταβλητή ελέγχου και προσδιορίζει την αρχική τιμή του μετρητή
- Τελική έκφραση είναι μια έκφραση ίδιου τύπου με τη μεταβλητή ελέγχου και προσδιορίζει την τελική τιμη του μετρητή

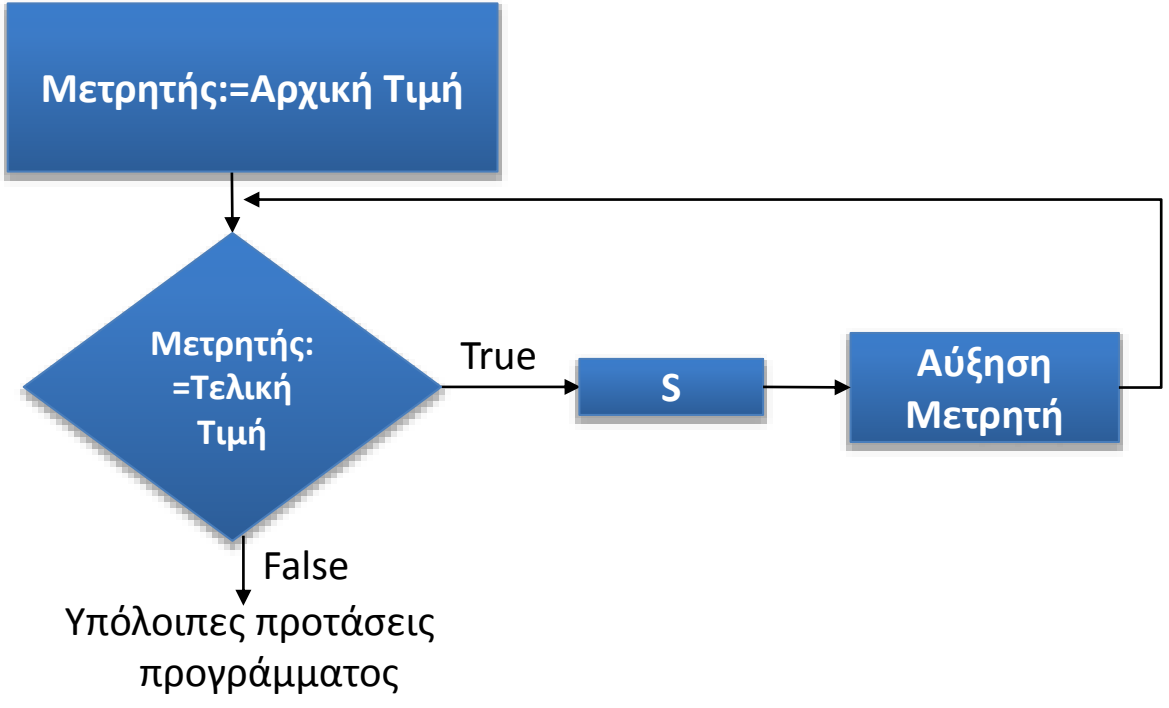
```
for x:=1 to 10 do writeln(x);
```

```
for x:=1 to 10 do writeln(x);
for c:='a' to 'e' do writeln(c);
for x:=3 to 1 do writeln(x);

                              end;
```

Η εντολή For 3/6

- Η ροή εκτέλεσης της εντολής For



Η εντολή For$_{4/6}$

- **Εξηγήσεις:** Στη δεύτερη σύνταξη της εντολής **downto** η τιμή του μετρητή ελαττώνεται από την τιμή της αρχικής έκφρασης έως την τιμή της τελικής έκφρασης:

```
for x:=10 downto 1 do writeln(x);
for c:='e' to 'a' do writeln(c);
for x:=1 to 3 do writeln(x);
```

- Αν η τιμή της αρχικής έκφρασης είναι ίση με την τιμή της τελικής έκφρασης και στους δύο τρόπου σύνταξης της εντολής, η πρόταση εκτελείται μια φορά:

```
for x:=6 to 6 do writeln(x);
for x:=6 downto 6 do writeln(x);
```

Η εντολή For 5/6

- **Παρατηρήσεις:** Η εντολή for χρησιμοποιείται όταν ο αριθμός των επαναλήψεων είναι προκαθορισμένος σε αντίθεση με τις άλλες εντολές επανάληψη while και repeat
- Με την εκτέλεση της εντολής αρχικά υπολογίζονται οι τιμές των εκφράσεων και καθορίζεται ο αριθμός των επαναλήψεων (έτσι μεταβολή των εκφράσεων δεν προκαλεί διαφοροποίηση του αριθμού των επαναλήψεων)

Επαναλήψεις=Τελική Εκφραση-Αρχική Εκφραση+1 (ένδειξη:to)
Επαναλήψεις=Αρχική Εκφραση-Τελική Εκφραση+1 (ένδειξη:downtoto)

Η εντολή For 6/6

- **Παράδειγμα:**
  Υπολογισμός του αθροίσματος $1^1+2^2+3^3+...N^n$

```
program total;
var N,i,x,sum,dyn: integer;
begin
readln(N);
sum:=0;
for  x:=1 to N do begin
                    dyn:=1;
                    for i:=1 to x do dyn:=dyn*x;
                    sum:=sum+dyn;
                    end;
writeln('Athroisma=', sum);
 end.
```

Ανακεφαλαίωση Εντολών Επανάληψης$_{1/2}$

- Οποιαδήποτε επαναληπτική εκτέλεση προτάσεων μπορεί να υλοποιηθεί και τις τρεις εντολές που προσφέρει για αυτό το σκοπό η Pascal
- Η εντολή for χρησιμοποιείται όταν ο αριθμός των επαναλήψεων είναι δεδομένος και δεν καθορίζεται από κάποια συνθήκη
- Η εντολή while χρησιμοποιείται όταν απαιτείται ο έλεγχος μια συνθήκης πριν την εκτέλεση της πρότασης (αντίθετα η εντολή repeat χρησιμοποιείται στις περιπτώσεις όπου απαιτείται η εκτέλεση μια πρότασης πριν το έλεγχο μια συνθήκης
- Η εντολή repeat ενδείκνυται στις περιπτώσεις όπου οι τιμές των μεταβλητών που καθορίζουν την τιμή της λογικής έκφρασης είτε είναι απροσδιόριστες ή δεν έχουν τη σωστή τιμή πριν την πρώτη εκτέλεσή της πρότασης

Ανακεφαλαίωση Εντολών Επανάληψης$_{2/2}$

- **Παράδειγμα:** Να γραφεί πρόγραμμα που να διαβάζει Ν αριθμούς και να υπολογίζει το άθροισμά τους. Να γίνει και με τις τρεις εντολές

```
program athr1;
var     N,num,counter,sum:
integer;
begin
read(N);
sum:=0;
counter:=1;
for  counter:=1 to N do begin
                        read(num);
                        sum:=sum+
                        num;
                        end;
writeln('Athroisma=', sum);
 end.
```

```
program athr2;
var     N,num,counter,sum:
integer;
begin
read(N);
sum:=0;
counter:=1;
repeat
read(num);
sum:=sum+num;
counter:=counter+1;
until counter>M
writeln('Athroisma=', sum);
 end.
```

```
program athr2;
var     N,num,counter,sum: integer;
begin
read(N);
sum:=0;
counter:=1;
while counter<=N do
                begin
                read(num);
                sum:=sum+num;
                Counter:=counter+1
                end;
writeln('Athroisma=', sum);
 end.
```

Έλεγχος Επανάληψης με Μεταβλητές$_{1/2}$

- Όταν η συνθήκη τερματισμού μια εντολής επανάληψης while ή repeat είναι πολύπλοκη, μπορούμε να αντικαταστήσουμε με μια μεταβλητή τύπου boolean
- Οι μεταβλητές αυτές συχνά ονομάζονται ‘σημαίες’ (flags)

Έλεγχος Επανάληψης με Μεταβλητές$_{2/2}$

- **Παράδειγμα:**
  Να γραφεί το πρόγραμμα το οποίο να διαβάζει ένα θετικό ακέραιο αριθμό και να τυπώνει (αν υπάρχει) ένα διαιρέτη του

```
program first_divisor;
var num,divisor: integer;
    flag: boolean;
begin
read(num);
divisor:=2;
flag:=false;
while not flag do begin
                        flag:=num mod divisor=0;
                        divisor:=divisor+1;
                        end;
 write('o arithmos', num);
 if flag then write('exei ena diaireth ton:', divisor-1)
         else write('den exei kanena diaireth');
 end.
```

Φωλιασμένες Εντολές Επανάληψης$_{1/2}$

- Η πρόταση που εκτελείται επαναληπτικά από μια εντολή επανάληψης μπορεί να είναι μια οποιαδήποτε πρόταση αποδεκτή από την Pascal
- Συνεπώς μπορεί να είναι μια άλλη εντολή επανάληψης δημιουργώντας έτσι φωλιασμένες εντολές επανάληψης

Φωλιασμένες Εντολές Επανάληψης$_{2/2}$

- Παραδείγματα φωλιασμένων εντολών επανάληψης αποτελούν οι ακόλουθες 4 προτάσεις, οι οποίες ας σημειωθεί εκτελούνται 12 φορές και παράγουν το ίδιο αποτέλεσμα:

```
1) x:=1;
while x<=3 do begin
                y:=2;
                while y<=5 do begin
                writeln(x,y);
                y:=y+1;
                end;
                x:=x+1;
                end;
```

```
2) x:=1;
repeat
y:=2;
    repeat
    writeln(x,y);
    y:=y+1;
    until y>5;
x:=x+1;
until x>3;
```

```
3) x:=1;
while x<=3 do begin
        y:=2;
        repeat
        writeln(x,y);
        y:=y+1;
        until y>5;
x:=x+1;
end;
```

```
4) for x:=1 to 3 do
        for y:=2 to 5 do;
          writeln(x,y);
```

Ασκήσεις$_{1/6}$

- **Παράδειγμα:**

Να γραφεί πρόγραμμα που να τυπώνει κάθετα τους αριθμούς από το 1 ως το 10 **(for)**

Εδώ έχουμε την μεταβλητή count (τύπου integer) της όποιας δίνουμε αρχική τιμή 1 και τελική τιμή 10 (άρα ο βρόγχος μας θα εκτελεστεί 10 φορές), αυτό που γίνετε μέσα στον βρόγχο είναι ότι τυπώνεται η τιμή που παίρνει κάθε φορά η μεταβλητή count δηλαδή 1,2,3,4,5,6,7,8,9,10.

```
program example1;
var count :integer;
begin
for count := 1 to 10 do begin
                         writeln(count);
                         end;
readln;
end.
```

Ασκήσεις$_{2/6}$

- **Παράδειγμα:**

Να γραφεί πρόγραμμα που να τυπώνει κάθετα τους αριθμούς από το 10 ως το 1 (**for**)

Κάνει το ίδιο με το example1 αλλά αυτή την φορά ο μετρητής count μετρά ανάποδα (downto) Άρα η μεταβλητή count παίρνει τις τιμές: 10,9,8,7,6,5,4,3,2,1.

```
program example2;
var count :integer;
begin
for count := 10 downto
1 do
begin
writeln(count);
end;
readln;
end.
```

Ασκήσεις$_{3/6}$

- **Παράδειγμα:**

Να γραφεί πρόγραμμα που να τυπώνει κάθετα τους αριθμούς από το 0 ως το 9 (**while**)

Καταρχάς δίνουμε αρχική τιμή στη μεταβλητή a:=0;. Όσο το a είναι μικρότερο του 10 τυπώνει το a και στη συνέχεια προσθέτει 1 … Άρα το a θα παίρνει τις τιμές 0,1,2,3,4,5,6,7,8,9. Στο a<10 (δηλαδή το 9) θα τερματιστεί ο βρόγχος για τον λόγω ότι **ΔΕΝ** θα ισχύει η συνθήκη και στην συνέχεια θα τερματιστεί το πρόγραμμα.

```
program example3;
var a :integer;
begin
a := 0;
while a < 10 do begin
        writeln (a);
        a := a + 1;
        end;
readln;
end.
```

Ασκήσεις$_{4/6}$

- **Παράδειγμα:**

Να γραφεί πρόγραμμα που να τυπώνει κάθετα τους αριθμούς από το 0 ως το 9 (**repeat**)

Όπως και στο example3 δίνουμε αρχική τιμή στη μεταβλητή a :=0;. Θα προσθέτει 1 στο a μέχρι το a να πάρει τιμή μεγαλύτερη του 9 Άρα το a θα παίρνει τις τιμές 0,1,2,3,4,5,6,7,8,9. Μόλις το a πάρει την τιμή 10 (a >9) θα τερματιστεί ο βρόγχος για τον λόγω ότι **ΔΕΝ** θα ισχύει η συνθήκη και στην συνέχεια θα τερματιστεί το πρόγραμμα

```
program example4;
var a :integer;
begin
a := 0;
repeat
writeln(a);
a := a + 1;
until (a > 9);
readln;
end.
```

Ασκήσεις$_{5/6}$

- **Παράδειγμα:**

Να γραφεί πρόγραμμα που να διαβάζει ένα σύνολο αριθμών και να τυπώνει το μικρότερο και το μεγαλύτερο. Το τέλος του συνόλου των αριθμών θα δηλώνεται με κάποιον συγκεκριμένο αριθμό (π.χ. -999).

```
program min_max;
var     x, min, max: integer;
begin
write('x');readln(x);
min:=x; max:=x;
while x<>-999 do begin
                        if x<min then min:=x else
                        if x>max then max:=x;
                        write('x=');readln(x)
                        end;
writeln('min=',min,'max=', max)
end.
```

Ασκήσεις$_{6/6}$

- **Παράδειγμα:**

Υπολογισμός του αθροίσματος:
H(N)=1+1/2+1/3+…+1/N

```
program Hsum;
var k,N: integer;
      h:real;
begin
write('N=');readln(N);
h:=0;
for k:=1 to N do h:=h+1/k;
writeln('H=',h)
end.
```

Βιβλιογραφία

Βλαχάβας Ι. (1994). Η γλώσσα προγραμματισμού Pascal. Εκδόσεις Γαρταγάνης Διονύσιος.

Κάβουρας Ι.Κ. (1999). Δομημένος Προγραμματισμός με Pascal. Εκδόσεις Κλειδάριθμος.

Αλεβίζου Θ., & Καμπουρέλης Α. (1995). Μαθήματα Προγραμματισμού: Εισαγωγή με τη Γλώσσα Pascal. Εκδόσεις Παπασωτηρίου.

Cooper D. (1993). Oh! Pascal!, An Introduction to Computing, του. Εκδόσεις Norton.

Larry R.N. (1998). Advanced Programming in Pascal with Data Structures. Εκδόσεις Macmillan USA.

Τσελίκης Γ.Σ., Τσελίκας Ν.Δ. (2012). C: από τη Θεωρία στην Εφαρμογή (Β' Έκδοση). Εκδόσεις Παπασωτηρίου.

Aho A.V., Hopcroft J.E., & Ullman J.D. (1974). The design and analysis of computer algorithms. Εκδόσεις Addison Wesley.

Abelson H., Sussman G.J., Sussman J. (1985). Structure and Interpretation of Computer Programs, MIT Press, McGraw Hill Book Company.

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

[1] http://creativecommons.org/licenses/by-nc-nd/4.0/deed.el

Τέλος Ενότητας

Επεξεργασία: Ευάγγελος Καρβούνης

Άρτα, 2015

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
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Τέλος Ενότητας

Εντολές Επανάληψης

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
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης
ΕΣΠΑ 2007-2013
Πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ