ΕΝΟΤΗΤΑ 1
ΣΥΜΒΟΛΑ ΚΑΙ ΣΤΟΙΧΕΙΑ ΤΗΣ PASCAL

Κάθε πρόγραμμα, που ειναι γραμμένο σε γλώσσα Pascal, αποτελείται από κάποια θεμελιώδη βασικά δομικά και συντακτικά στοιχεία.Αυτά είναι τα εξής :

- **Αλφάβητο**
- **Ονόματα**
- **Αριθμοί**
- **Ακολουθίες χαρακτήρων**

Ένα πρόγραμμα, λοιπόν, θα πρέπει πρώτα από όλα να αρχίζει με την *επικεφαλίδα* ,η οποία περιέχει. το όνομα που αποδίδεται στο πρόγραμμα συνοδευμένο με ένα σύνολο παραμέτρων που χρησιμοποιούνται από τον Compiler κατά τη διάρκεια της μεταγλώττισης. Η γενική του μορφή είναι :

Program όνομα ;

Παράδειγμα:

```
Program test ;
```

Αριθμοί

Για τους αριθμούς χρησιμοποιείται η δεκαδική μορφή.
Οι αριθμοί στην Pascal χωρίζονται σε δύο τύπους οι οποίοι είναι :

- **Ακέραιοι**, που συνήθως είναι από -maxint+1...maxint, όπου maxint είναι μια σταθερά που εξαρτάται από την ακρίβεια του συστήματος.
  Παράδειγμα : 31.
- **Πραγματικοί**, οι οποίοι αναπαρίστανται με δύο τρόπους :
  1) x.y
  2) x.y E +/- m , όπου x,y σειρά ψηφίων.
  Παραδείγματα : 3,14 και 2.5E-2.

Ακολουθίες χαρακτήρων

Μία ακολουθία χαρακτήρων ή string ορίζεται ως εξής :

<string> = ' < χαρακτήρας > { <χαρακτήρας>}'

Παράδειγμα:

```
writeln('Δώσε μου τον αριθμό'),
```

όπου το περιεχόμενο ανάμεσα στους μονούς αποστρόφους ', είναι ένα string.
Τα strings χρησιμοποιούνται ευρέως για την εξαγωγή μηνυμάτων και επεξεργασία αλφαριθμητικών δεδομένων.

Ονόματα

Τα ονόματα ή προσδιοριστές (identifiers) προσδιορίζουν προγραμματιστικές οντότητες, δηλ. μεταβλητές, υποπρογράμματα, τύπους δεδομένων κ.λ.π. Στη Pascal ένα όνομα ειναι μια ακολουθία αλφαριθμητικών χαρακτήρων , η οποία αρχίζει υποχρεωτικά από ένα γράμμα. Μπορούν να

χρησιμοποιούν αδριάκριτα κεφαλαία ή μικρά γράμματα αναμεμιγμένα ή όχι. Σε πολλές εκδόσεις χρησιμοποιείται και ο χαρακτήρας "_"(underscore). Η δομή τους έχει ως εξής :

id := (γράμμα) {(γράμμα),|(ψηφίο),|"_"}.

Τα ονόματα χωρίζονται σε δύο κατηγορίες :

- *Predifined* δηλ. προκαθορισμένα. Είναι αυτά που είναι ήδη καθορισμένα από τη γλώσσα και το σύστημα. Παραδείγματα :
  maxint, write, integer.
- *User defined* δηλ. ονόματα που καθορίζει ο ίδιος ο προγραμματιστής. Παραδείγματα :
  filename1, quee.

ΣΥΜΒΟΥΛΗ

- *Χρησιμοποιείτε ονόματα που θα σας διευκολύνουν να θυμηθείτε τη λειτουργία του προγράμματος ύστερα από καιρό.*

Αλφάβητο

Το αλφάβητο περιέχει όλους τους χαρακτήρες που χρησιμοποιούνται για το σχηματισμό των λέξεων και των προτάσεων της γλώσσας.
Τα στοιχεία που αποτελούν λοιπόν το αλφάβητο της γλώσσας Pascal είναι τα εξής:

Κεφαλαία γράμματα : **A, B, C, D,..., Z**
Μικρά γράμματα : **a, b, c ,d ,..., z**
Δεκαδικά ψηφία : **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
Ειδικοί Χαρακτήρες : **+ - * / < > + ( ) [ ] . , ; : ' ^ _**

Δομή του προγράμματος

Ένα τυπικό πρόγραμμα σε Pascal είναι δηλωμένο σε έξι τμήματα:
Πέντε δηλωτικά τμήματα, που συνιστούν το συνολικό δηλωτικό τμήμα του κώδικα, και ένα εκτελέσιμο τμήμα.

Τα δηλωτικά τμήματα προηγούνται του τμηματος των εκτελέσιμων εντολών και έχουν μια συγκεκριμένη διάταξη, εισάγονται δε με μια ενδεικτική δεσμευμένη λέξη της γλώσσας:

- Τμήμα δηλώσεων επιγραφών : **Label**.....;
- Τμήμα ορισμού σταθερών: **Const**.....;
- Τμήμα ορισμού τύπων: **Type**.....;
- Τμήμα δηλώσεων μεταβλητών: **Var**.....;
- Τμήμα δηλώσεων διαδικασιών ή συναρτήσεων : **Procedure** ή **Function**....;
- Τμήμα εκτελέσιμων εντολών: **Begin.....End**.

Η εμφάνιση των δηλωτικών τμημάτων είναι προαιρετική, ενώ του εκτελέσιμου προφανώς υποχρεωτική. Όλα όμως τα αντικείμενα που χρησιμοποιούνται στο εκτελέσιμο μέρος είναι υποχρεωτικό να δηλώνονται.

ΕΝΟΤΗΤΑ 2
ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΑΞΕΙΣ

Για να συνθέσετε προγράμματα σε μία γλώσσα προγραμματισμού θα πρέπει πρώτα να εξοικειωθείτε με τους κανόνες που αφορούν τα δεδομένα καθώς και με τις πράξεις που τα αφορούν. Εδώ υπεισέρχονται οι τύποι δεδομένων.

Ορισμός :

- **Τύπος δεδομένων** είναι ένα σύνολο αντικειμένων ή τιμών και ένα σύνολο λειτουργιών ή πράξεων πάνω σε αυτά τα αντικείμενα.

  Οι τιμές ενός τύπου δεδομένων αποτελούν το *πεδίο ορισμού* του. Ένας τύπος δεδομένων εξασφαλίζει ένα είδος αφαίρεσης. Μετάβαση από τις λεπτομέρειες αναπαράστασης ενός δεδομένου στο σύνολο των δεδομένων. Για παράδειγμα :

  Το σύνολο των ακεραίων {-3, -2, -1, 0, 1, 2, 3} αναπαριστάται ως : Integer = { I, +, -, *}.

  Οι τύποι δεδομένων χωρίζονται σε δύο είδη.Αυτοί είναι :
- **Basic ή βασικοί τύποι** που είναι προκαθορισμένοι από τη γλώσσα και είναι:
  - Ακέραιος (Integer)
  - Πραγματικός (Real)
  - Χαρακτήρας (Char)
  - Λογικός (Boolean)
  - Τύποι ορισμένοι από τον χρήστη.
- **Σταθερά**, είναι η πραγματική τιμή ενός δοσμένου τύπου δεδομένων.

  Η σταθερά δηλώνεται με την εντολή - δήλωση CONST.

Ακολουθούν μερικά παραδείγματα τύπων δεδομένων.

Για τύπο integer : k= 12 , l =3
Για τύπο real : j= 27,34, lock= 13,987
Για τύπο char : blank= ' '
Για τύπο boolean : error= false
Στη συνέχεια, οι σταθερές χρησιμοποιούνται για την ανάθεση τιμών σε μεταβλητές μέσα στο πρόγραμμα.

Βασικοί τύποι

Οι basic ή βασικοί τύποι είναι προκαθορισμένοι από τη γλώσσα και είναι οι εξής :

- Ακέραιος (Integer)
- Πραγματικός (Real)
- Χαρακτήρας (Char)
- Λογικός (Boolean)

Ακέραιος - Integer

Οι ακέραιοι στην Pascal ακολουθούν το μαθηματικό πρότυπο.Έτσι οι τελεστές ακεραίων είναι οι εξής :

Πράξη	Τύπος ορίσματος1	Τύπος ορίσματος2	Τύπος αποτελέσματος
+	integer	integer	integer

-	integer	integer	integer
*	integer	integer	integer
/	integer	integer	real
mod	integer	integer	integer

Ενώ οι τελεστές συσχέτισης :

=	ίσο
<>	διάφορο
>	μεγαλύτερο
<	μικρότερο
>=	μεγαλύτερο ή ίσο
<=	μικρότερο ή ίσο

Οι ακέραιοι διαθέτουν και κάποιες τυποποιημένες συναρτήσεις.

Πραγματικός - Real

Οι πραγματικοί όπως και οι ακέραιοι ακολουθούν το μαθηματικό πρότυπο.
Οι τελεστές των πραγματικών είναι :

Τελεστής	Τύπος Ορίσματος	Τύπος Αποτελέσματος
+	Real ή Integer	Real ή Integer
-	Real ή Integer	Real ή Integer
*	Real ή Integer	Real ή Integer
/	Real ή Integer	Real

Χαρακτήρας - Char

Τελεστές και συναρτήσεις χαρακτήρων
Είναι :

Ord(ch) Διατακτική συνάρτηση.Δίνει την εσωτερική τιμή του ch.
Chr(Int) Συνάρτηση χαρακτήρα.Δίνει τον ακέραιο χαρακτήρα που αντιστοιχεί στον Int.
Pred(ch) Συνάρτηση προηγούμενου.Δίνει τον προηγούμενο διατεταγμένο χαρακτήρα του ch.
Succ(ch) Συνάρτηση διαδόχου. Δίνει τον επόμενο χαρακτήρα του ch.

Παραδείγματα:

```
Pred('C') = 'B'
Succ('X') = 'Y'
Ord('H') = Ord('I') - 1
Ord('X') = Ord('Y') - 1
```

Λογικός Τύπος - Boolean

Ο τύπος boolean μπορεί να πάρει δύο τιμές

true (αληθής) και
false (ψευδής)

Για παράδειγμα:

4 = 5 είναι false
11 = 11 είναι true
'C' > 'B' είναι true
2,3 < -7,2 είναι false

Ο τύπος αυτός χρησιμοποιείται σε λογικές εκφράσεις και σε υπολογισμούς που καθορίζουν λήψεις αποφάσεων σε ένα πρόγραμμα.Χρησιμοποιείται δηλαδή ως "flag" (σημαία), που η τιμή της καθορίζει την έκβαση του προγράμματος.

Παράδειγμα για κατανόηση του flag

Το παρακάτω πρόγραμμα δείχνει τη σημασία του flag.

```
Program convert;
Const
        zero = '0'; nine = '9';
        teleia = '.';
        epsilon = 1E-12;
Var
        apot : real;
        dek : integer;
        ch : char;
         telos : boolean; {Εδώ δηλώνουμε την μεταβλητή που λειτουργεί ώς flag   και σηματοδοτεί το τέλος του
προγράμματος}

Begin {main}
  telos:=false; {Αρχικοποιούμε την boolean τιμή του flag }
  While Not telos Do {Ελέγχουμε την τιμή της μεταβλητής flag }
  Begin
        apot:=0;
        Repeat read(ch);
        Until (zero <= ch) And (ch <= nine);
        While (zero <= ch) And (ch <= nine) Do
        Begin
                apot := 10*apot+ord(ch)-ord(zero);
                read(ch)
        End; {while}
        If ch = teleia Then
                Begin
                        dek:=0; read(ch);
                        While (zero <= ch) And (ch <= nine) Do
                        Begin
                                apot := 10*apot+ord(ch)-ord(zero);
                                read(ch); dek := dek+1;
                        End; {while}
                        While dek > 0 Do
                        Begin
                                apot := apot/10; dek := dek-1
                        End {while}
                End; {if}
        WriteLn(apot); ReadLn;
        If apot < epsilon Then telos := true {Αν ισχύει η συνθήκη αλλάζουμε την τιμή του flag και έτσι βγαίνουμε από τον
βρόγχο}
  End; {while}
  Write('* end of execution *')
End. {convert}
```

Λογικοί Τελεστές

Οι λογικοί τελεστές που χρησιμοποιούνται από την Pascal είναι :

- **AND** λογικό **ΚΑΙ**
- **OR** λογικό **Η**
- **NOT** λογικό άρνηση

Λογικές Συναρτήσεις

Οι τρεις σημαντικότερες λογικές συναρτήσεις που παίρνουν μία λογική τιμή TRUE ή FALSE, που είναι οι εξής :

ODD(Int), που είναι True όταν ο Int είναι περιττός και False όταν είναι άρτιος.
Eof = End Of File: χρησιμοποιείται για να δηλώσει αν έχει τελειώσει τη διαδικασία εισόδου δεδομένων. Παίρνει την τιμή true όταν έχει τελειώσει και false στην αντίθετη περίπτωση.
Eoln = End Of Line χρησιμοποιείται για να δηλώσει αν μία γραμμή έχει τελειώσει τη διαδικασία δεδομένων εισόδου. Παίρνει την τιμή true όταν έχει τελειώσει και false στην αντίθετη περίπτωση.

Τυποποιημένες συναρτήσεις

Οι τυποποιημένες συναρτήσεις της Pascal είναι :

Τιμή\'Ορισμα	integer	real	boolean	char	file
integer	pred\succ\abs\sqr	trunc\round	ord	ord	-
real	sin\cos\arctan\ln\exp\sqrt	abs\aqr\sin\cos\arctan\ln\exp\sqrt	-	-	-
boolean	odd	-	pred\succ	-	-
char	chr	-	-	pred\succ	-

Τύποι οριζόμενοι απο τον χρήστη

Οι οριζόμενοι από τον χρήστη τύποι (user defined) μπορούν να ορίζονται είτε κατευθείαν στο τμήμα δηλώσεων μεταβλητών του προγράμματος, είτε στο τμήμα ορισμού τύπων. Οι τύποι που ορίζονται από τον χρήστη είναι :

- **Τύποι απαρίθμησης (enumerated types)**.
  Καθορίζονται από ένα διατεταγμένο σύνολο τιμών, οι οποίες υποδηλώνονται από ονόματα.
  Ορίζονται ως εξής:

  TYPE < όνομα τύπου > = (< όνομα > {,<όνομα >})

- **Τύποι διαστήματος (Subrange types)**
  Ορίζονται από ένα ήδη ορισμένο βαθμωτό τύπο σαν ένα υποδιάστημα διαδοχικών τιμών του.
  Δηλώνονται ως εξής :

  TYPE < όνομα τύπου > = <σταθερά> ...<σταθερά> ;

- **Δομημένοι τύποι**
  Ορίζονται σαν σύνθεση απλούστερων τύπων:

  - Τύποι συνόλων(sets)
  - Πίνακες (arrays)
  - Εγγραφές (records)

- Δείκτες.
- Αρχεία.

ΕΝΟΤΗΤΑ 3
ΒΑΣΙΚΕΣ ΕΝΤΟΛΕΣ

Οι βασικές εντολές της Pascal χωρίζονται σε εντολές εισόδου - εξόδου, σε εντολές ανακύκλωσης και εκχώρησης και σε εντολές απόφασης.

Οι εντολές εισόδου - εξόδου είναι :

- Read - Readln
- Write - Writeln

Οι εντολές ανακύκλωσης ή δημιουργίας βρόγχου σε ένα πρόγραμμα :

- While...do
- Repeat..until
- For

Επίσης, υπάρχουν και οι εντολές επιλογής που η λειτουργία τους καθορίζεται στη λήψη λογικών αποφάσεων οι οποίες είναι :

- If
- Case

Όλες οι παραπάνω εντολές μαζί με τις εντολές ανάθεσης αποτελούν τη βάση των γλωσσικών δομών της Pascal. Πιο πολύπλοκες δομές θα εξεταστούν σε παρακάτω ενότητες.

Εντολές Εισόδου - Εξόδου

Οι διαδικασίες *εισόδου* στην Pascal είναι οι εξής :

1. **Read**, η οποία διαβάζει το επόμενο στοιχείο από την είσοδο και
2. **Readln**, η οποία κάνει την ίδια λειτουργία αλλά την επόμενη φορά που θα εμφανιστεί εντολή Read ή Readln στο πρόγραμμα ο compiler διαβάζει τα στοιχεία της επόμενης γραμμής. Με άλλα λόγια δηλαδή η εντολή Readln, διαβάζει γραμμές δεδομένων από την είσοδο.

Η γενική μορφή τους είναι :

Read (f, v1, v2,...,vn) , όπου f είναι το όνομα του αρχείου εισόδου v1,v2,...,vn είναι ονόματα μεταβλητών τύπου integer, real, char, και όχι boolean.

Π.χ. `Read(a), Readln(ch)`

Οι διαδικασίες εξόδου στην Pascal είναι οι εξής :

1. **Write**, που εκτυπώνει χαρακτήρες στην οθόνη και
2. **Writeln**, που κάνει το ίδιο αλλά την επόμενη φορά που θα βρεθεί Write ή Writeln στο πρόγραμμα, τότε η εκτύπωση χαρακτήρων αρχίζει από την επόμενη γραμμή.

Η γενική τους μορφή είναι:

Write (f,a1,a2,a3,..,an) , ο΄που f μια προαιρετική παράμετρος και δηλώνει το αρχείο εξόδου και τα
a1,a2,a3,..,an είναι παράμετροι εξόδου.

Π.χ.

```
write(b), writeln(ch)
```

Το πρόγραμμα αυτό ζητά και εκτυπώνει δεδομένα.

```
PROGRAM arithmetic;
VAR
  a,b:integer;
BEGIN
  WRITE('Enter the number a=');
  READLN(a);
  WRITELN('number',a:4,'entered');
  WRITELN('Enter the number b=');
  READLN(b);
  WRITELN('number',b:4,'entered');
  WRITELN('The results are:',a+b:5,a-b:5,a*b:5,aDivb:5)
END.
```

Εντολές Ανακύκλωσης

While...Do

Η εντολή While..do της Pascal είναι από τις πιο βασικές εντολές της γλώσσας. Στόχος της είναι να δημιουργήσει ένα βρόχο μέσα στο πρόγραμμα.
Η σύνταξη είναι :

While < λογική έκφραση είναι True> do <εντολή>

Όσο η λογική έκφραση παραμένει αληθής, η εντολή ή η ομάδα εντολών που ακολουθεί το do θα εκτελείται.Όταν αυτή γίνει ψευδής η εκτέλεση των εντολών σταματά. Για τις λογικές αυτές εκφράσεις είναι προτιμότερη η χρήση των **flags** (boolean μεταβλητές).

Παράδειγμα:

```
WHILE a<6 DO
BEGIN
WRITE(a);
WRITE(a+1);
END;
```

Το ακόλουθο πρόγραμμα τυπώνει τους αριθμούς 1,2,3,4,5,6 χρησιμοποιώντας την εντολή while...do.

```
PROGRAM numbers;
VAR
  NUMBER:integer;
BEGIN
  number:=0;
  WHILE number <=5 Do
  BEGIN number:="number+1;"
    WRITE(number,' ')
  END
END.
```

Repeat...Until

Στην εντολή Repeat..until, όπως και στην εντολή While...do, χρησιμοποιούμε **flags** για να δημιουργήσουμε το βρόχο λειτουργίας της εντολής.
Γενική μορφή :

Repeat <εντολή>{; <εντολή> } Until <Λογική έκφραση>

Μεταξύ των δεσμευμένων λέξεων Repeat και Until υπάρχουν μία ή περισσότερες εντολές και η συνθήκη ορίζεται με λογική έκφραση.

Ένα παράδειγμα της εντολής είναι :

REPEAT read(ch);write(ch) UNTIL ch='*';
που θα διαβάσει μία σειρά χαρακτήρων συμπεριλαμβανομένου και του τελικού χαρακτήρα '*'.

Το πρόγραμμα υπολογίζει και τυπώνει την ποσότητα h=1+1/2+...+1/n

```
PROGRAM crepeat;
VAR
  i,n:integer;
  h:real;
BEGIN
  WRITE('Enter the number :');
  READLN(n);
  WRITE(n);
  h:=0; {initialization}
  i:=n;
  REPEAT
    h:=h+1/i;
    i:=i-1
  UNTIL i=0;
  WRITELN(h)
END.
```

ΣΥΜΒΟΥΛΗ

- *Οι συνθήκη πρέπει να είναι όσο το δυνατόν απλούστερη αφού υπολογίζεται κάθε φορά που εκτελείται ο βρόχος.*

For

Η εντολή For ακολουθεί το ίδιο πρότυπο με τις εντολές while..do και repeat..until και συντάσσεται ως εξής :

FOR <μεταβλητή καταμέτρησης> : = <έκφραση1> TO <έκφραση2> DO <εντολή> *ή*
FOR <μεταβλητή καταμέτρησης> : = <έκφραση1> DOWNTO <έκφραση2> DO <εντολή>

Με αυτόν τον τύπο εντολής επανάληψης καθορίζεται κατά την είσοδο πόσες φορές πρόκειται να εκτελεστεί ο βρόχος. Προαιρετικά μπορεί να χρησιμοποιηθεί η έκφραση Downto αντί του To .

Ενα παράδειγμα που τυπωνει σε διαφορετικές γραμμές τους ακεραίους από το 1 εώς το 10:
```
FOR i:=1 TO 10 DO WRITELN(i);
```

Το ακόλουθο πρόγραμμα βρίσκει το μεγαλύτερο αριθμό μεταξύ n ακεραίων από την είσοδο.

```
Program maximum;
{ This program finds and prints the biggest integer among n integers }
Var
  n,number,i,max:integer;
```

```
Begin
  Write('How many numbers do you want to enter ? ');
  ReadLn(n);
  WriteLn('Enter the numbers ->');
  Write('number',1:3,':');
  ReadLn(number);
  max:=number;
  If n >=2 Then
  For i:=2 To n Do
  Begin
    Write('number',i:3,':');
    ReadLn(number);
    If max < number Then max := number
  End;
  WriteLn('maximum:',max:8)
End.
```

If

Η εντολή If..then είναι η πιο χαρακτηριστική περίπτωση εντολής υπό συνθήκη. Χρησιμοποιείται για τη λήψη αποφάσεων όταν μία μεταβλητή παίρνει πολλές τιμές σε ένα πρόγραμμα. Συντάσσεται ως εξής :

<εντολή IF>::= IF <λογική έκφραση> THEN <εντολή>
| IF <λογική έκφραση> THEN <εντολή> ELSE <εντολή>

Η εντολή ELSE είναι προαιρετική και εφαρμόζεται σε περιπτώσεις που θέλουμε να υλοποιήσουμε περισσότερες από μία αποφάσεις

Ένα παράδειγμα χρήσης της IF είναι το παρακάτω :

```
IF number>9 THEN i:=number + 1 ELSE i:=number - 1
```

Το παρακάτω πρόγραμμα επιλύει μία δευτεροβάθμια εξίσωση.

```
PROGRAM solve;
VAR
  a,b,c,d,prag,fant,riza1,riza2:real;
BEGIN
  WRITELN('solution of the equalization of the second degree axú+bx+c=0');
  READ(a,b,c);
  IF ((a=0) And (b=0)) THEN WRITELN('The equalazation has no solution ')
  ELSE IF a=0 THEN WRITELN('The equalization is of the first degree with solution=',-c/b:10:6)

  ELSE {class of second degree equalization}
  BEGIN
    prag:=-b/(2*a);
    d:=sqr(b)-4*a*c;
    fant:=sqrt(abs(d))/(2*a);
    IF d >0 THEN WRITELN('The equalization has real roots:',prag+fant:10:6,' □□  ', prag-fant:10:6)
    ELSE WRITELN('The equalization has complex roots:',prag,'+i*',fant,' □□ ',prag,'-i*',fant)
 END {else}
END. {solve}
```

Case

Η εντολή CASE..OF χρησιμοποιείται για την υλοποίηση πολλαπλών επιλογών. Η σύνταξη της έχει ως εξής :

CASE <έκφραση> OF
< λίστα σταθερών>: <εντολή>
{ ;<λίστα σταθερών> : <εντολή> }
END

Για παράδειγμα:

```
CASE Weekday OF
Monday : WRITELN(1);
Tuesday : WRITELN(2);
Friday : WRITELN(5);
```

Οι εντολές ανάθεσης ή εκχώρησης χρησιμοποιούνται από το πρόγραμμα όταν θέλουμε να αποδωθούν τιμές σε μία μεταβλητή με διαφορετικό τρόπο από αυτόν της εισόδου.
Η σύνταξη τους είναι :

<μεταβλητή> : = < έκφραση>

Βασικός κανόνας είναι ότι πρέπει πρώτα να υπολογιστεί η τιμή της έκφρασης και μετά να πραγματοποιηθεί η εκχώρηση.

Παραδείγματα :

```
temp : = a; a : = b; b : = temp;
found : = true;
bool : = (x>y) AND (x>10) OR found;
```

ΕΝΟΤΗΤΑ 4
ΔΙΑΔΙΚΑΣΙΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

Ο προγραμματιστής συχνά χρειάζεται να ονομάσει μία ομάδα ή ένα συγκρότημα εντολών που επιτελούν συγκεκριμένο σκοπό μέσα στο πρόγραμμα. Τα συγκροτήματα αυτά εντολών λέγονται υπορουτίνες ή υποπρογράμματα και χωρίζονται στα δύο εξής είδη :

- **Διαδικασίες**
- **Συναρτήσεις**

Η σημαντικότερη διαφορά ανάμεσα στις διαδικασίες και στις συναρτήσεις είναι ότι η συνάρτηση έχει μία τιμή που μπορεί να χρησιμοποιηθεί σε μία έκφραση, ενώ η διαδικασία δεν έχει τιμή που να αντιστοιχεί στο όνομα της.Δηλαδή η συνάρτηση χρησιμοποιείται σα μεταβλητή ενώ η διαδικασία σαν εντολή.

Η Διαδικασία ή Procedure είναι ένα μικρότερο κομμάτι προγράμματος από το συνολικό μέσα στο οποίο εκτελείται ένα σύνολο εντολών. Η τυπική σύνταξη μίας διαδικασίας είναι :

<εντολή διαδικασίας> := <όνομα διαδικασίας> | <όνομα διαδικασίας>(<πραγματική παράμετρος> {,<πραγματική παράμετρος>})

Παραδείγματα τέτοιων δηλώσεων είναι :

```
Compute; countchars;
```

Για να εκτελεστεί μία διαδικασία απλά αναγράφεται το όνομα της μέσα στο πρόγραμμα. Έτσι, το πρόγραμμα γίνεται πιο ευανάγνωστο.
Το παράδειγμα εξηγεί τα παραπάνω.

Η κλήση του ονόματος heading, εκτελεί την εξής procedure :

```
PROCEDURE heading
CONST width = 20 ;
VAR I : integer ;
BEGIN
  FOR I := 1 TO width DO write('*');
  writeln('*Heading*');
  FOR I:= 1 TO width DO write('*');
  writeln ;
end;
```

Και θα τυπωθεί :
****
Heading
****

Για να ενσωματωθεί η διαδικασία αυτή σε ένα πρόγραμμα θα δηλωνόταν :

```
PROGRAM print (input,output);
VAR .....
PROCEDURE heading;
BEGIN
  .......
  εντολές
  heading;
END.
```

Στο παράδειγμα για να εκτυπώσουμε 30 αστερίσκους θα έπρεπε να δηλώναμε σαν μεταβλητή το μήκος της σειράς και να το εκτελούσαμε με παράμετρο το 30.

Κάθε μεταβλητή ή τιμή που μεταβιβάζεται από ή προς το πρόγραμμα και αλληλεπιδρά με αυτό ονομάζεται παράμετρος. Οι παράμετροι δηλώνονται στην αρχή μίας διαδικασίας μέσα στον κατάλογο παμέτρων.Αυτές οι παράμετροι ονομάζονται τυπικές παράμετροι ή οποία υποκαθιστά τις πραγματικές παραμέτρους που παίρνει η διαδικασία από την είσοδο. Η επικεφαλίδα της procedure θα ήταν :

procedure heading(width:integer);

Εδώ το width είναι η τυπική παράμετρος και το πεδίο integer θα δηλώνει την πραγματική παράμετρο.

Το πρόγραμμα αυτό υλοποιεί το γνωστό αλγόριθμο του Ευκλείδη

```
PROGRAM fraction;
VAR
  i,num,den,numcopy,dencopy:integer;
  remainder,times:integer;
PROCEDURE lowterm;
BEGIN
  numcopy:=num;
  dencopy:=den;
  WHILE dencopy <>0 DO {find consecutive mods}
  BEGIN
    remainder:=numcopy Mod dencopy;
    numcopy:=dencopy;
    dencopy:=remainder
  END;{While}
{ Στο σημείο αυτό numcopy = ΜΚΔ(num,den) }
  IF numcopy >1 THEN {divide with the GCD}
  BEGIN
    num:=num Div numcopy;
    den:=den Div numcopy
  END
END; {lowtern}

BEGIN
  WRITE('Give the amount of the fractals : ');
  READLN(times);
  FOR i := 1 TO times DO
  BEGIN
    WRITE('Give the numerator ',i:2,' fractals :');
    READLN(num,den);
    lowterm; {procedure call}
    WRITELN(num,'/',den)
  END
END. {fraction}
```

Η συνάρτηση ή Function είναι ένα όνομα που δίνεται σε ένα συγκρότημα εντολών και μετά την εκτέλεση της, ανατίθεται σε αυτήν μία τιμή.

Η τυπική σύνταξη της είναι :

<επικεφαλίδα συνάρτησης> ::= Function<όνομα> : <τύπος αποτελέσματος> ;
| Function <όνομα> <λίστα παραμέτρων> : <τύπος αποτελέσματος> ;<τύπος αποτελέσματος>
::= <όνομα τύπου>.

Παραδείγματα:

```
Function max(a,b:integer):integer;
Function average (a,b:real):real;
```

Μία συνάρτηση θα πρέπει πάντα να παράγει ένα αποτέλεσμα και ο τύπος του πρέπει πάντα να καθορίζεται. Η τιμή της συνάρτησης δίνεται σε αυτήν μόλις εκτελεστεί. Δείτε το παράδειγμα .

```
Function MEAN(A,B:REAL):REAL;
Begin
  MEAN:=(A+B)/2.0
End;
```

Η συνάρτηση παίρνει κάποια τιμή και χρησιμοποιείται μέσα σε εκφράσεις. Έτσι, γράφοντας : Value := Mean (X,Y), με X,Y πραγματικές παράμετρους, θα εκτυπωθεί το X/Y /2.0.

Το παρακάτω πρόγραμμα υπολογίζει την τετραγωνική ρίζα.

```
PROGRAM square_root;
CONST
 epsilon=1E-6;
VAR
 x:real;

FUNCTION root(x,eps:real):real;
VAR
 a,g,temp:real;
BEGIN
 a:=x;g:=1; {Αρχικοποιήσεις}
 WHILE abs((a-g)/g) > eps DO
 BEGIN
  temp:=a;
  a:=(a+g)/2;
  g:=2*temp*g/(temp+g)
 END; {While}
 root:= a
 END; {root}

BEGIN {square_root}
 READLN(x);
 WRITELN(x,' square root=',root(x,epsilon) )
END. {square_root}
```

Πέρασμα Μεταβλητών με Τιμή και με Αναφορά.

Υπάρχουν δύο τρόποι για το πέρασμα μεταβλητών ή τιμών σε μία διαδικασία :

1.Πέρασμα με τιμή
2.Πέρασμα με αναφορά

Στην Pascal ορίζεται ότι οι διαδικασίες δηλώνονται μετά τις μεταβλητές. Εκτός από μεταβλητές και τιμές οι διαδικασίες μπορούν επίσης να δέχονται σαν παραμέτρους και διαδικασίες ή συναρτήσεις.

Πέρασμα με τιμή

Οι παράμετροι τιμής εισάγονται στον κατάλογο παραμέτρων χωρίς το σύμβολο VAR, και αντιστοιχούν σε πραγματικές παράμετρους που είναι γενικά εκφράσεις. Τότε λέμε ότι οι παράμετροι μεταβιβάζονται **με τιμή**. Ο μηχανισμός μεταβίβασης παραμέτρου με τιμή μπορεί να περιγραφεί ως εξής:

Κατά την κλήση μιας διαδικασίας και πριν από την είσοδο στο σώμα της, δεσμεύεται μία νέα περιοχή μνήμης για κάθε παράμετρο τιμής και αντιγράφονται εκεί οι πραγματικές παράμετροι. Η τιμή μιας μεταβλητής δεν επηρεάζεται μετά τον τερματισμό της διαδικασίας ενώ ο χώρος που δεσμεύτηκε για την αντιγραφή ελευθερώνεται.
Στο παράδειγμα η παράμετρος n είναι παράμετρος τιμής.

```
PROCEDURE find_fact(n:integer); {Υπολογισμός του n!}

VAR x,prod : integer
BEGIN
    prod:=1; x:=n;
    WHILE x<>0 DO
    BEGIN
        prod:=x*prod;
        x:=x-1;
    END;
    WRITELN(' ':10,n:3,'!=',prod)
END;
```

Πέρασμα με αναφορά

Όταν εισάγουμε σε μία διαδικασία μεταβλητές χρησιμοποιώντας πέρασμα με αναφορά τότε κάνουμε χρήση της VAR στον κατάλογο των παραμέτρων. Στο πέρασμα με αναφορά κατά την κλήση της διαδικασίας δίνονται σε αυτήν οι διευθύνσεις των πραγματικών παραμέτρων και αντιστοιχούνται σε αυτές οι τυπικές παράμετροι.Έτσι, υπάρχει αλληλεπίδραση ανάμεσα στις πραγματικές και στις τυπικές παραμέτρους.

Στο παράδειγμα οι παράμετροι A,B είναι μεταβλητές παράμετροι και αντιπροσωπεύουν τα δεδομένα τους.

```
Procedure square (VAR A,B:Real);
Begin
  A:=A*A;
  B:=B*B;
end;
```

Ακολουθεί ένα πρόγραμμα για να διαπιστωθεί η διαφορά στις δύο μεθόδους.

```
Program exch;
Var
  a,b:integer;

Procedure exchange(x, y : integer);
Var
  temp:integer;
Begin
  temp:= x;
  x:= y;
  y:= temp;
  WriteLn(x:2,y:2)
End; {exchange}

Begin {exch}
  ReadLn(a, b);
  exchange(a, b);
  WriteLn('values of actual parameters :',a:2,b:2)
```

End. {exch}
Εκτέλεση:

3 6<cr>
6 3
values of actual parameters : 3 6

Όπως φαίνεται καθαρά μετά την έξοδο από την υπορουτίνα exchange οι μεταβλητές α,β κρατούν την αρχική τους τιμή. Αν χρησιμοποιούσαμε μεταβλητές παράμετρους (VAR x,y:real) και εκτελούσαμε εκ νέου το πρόγραμμα θα λάβουμε:

3 6<cr>
6 3
values of actual parameters : 6 3

Η Pascal είναι μία block-structured γλώσσα. Έτσι, σε κάθε block υπάρχει εμβέλεια καθορισμένη για κάθε μεταβλητή. Στο παρακάτω παράδειγμα γίνεται αυτό αντιληπτό.

```
PROGRAM  A
procedure B
procedure D
Begin {D}
  ....
end; {D}
end; {B}
procedure C
procedure E
begin {E}
  ...
end; {E}
procedure F
begin {F}
  ..
end; {F}
begin {C}
  ..
end; {C}
begin {A}
  ..
end. {A}
```

Scope μεταβλητών

Μεταβλητές και που δηλώνονται	Εμβέλεια στα blocks
program A	A,B,C,D,E,F
procedure B	B,D
procedure C	C,E,F
procedure D	D
procedure E	E
procedure F	F

Από τον παραπάνω πίνακα βλέπουμε την εμβέλεια των μεταβλητών στο πρόγραμμα.

Η επίλυση προβλημάτων απλουστεύεται με την χρήση διαδικασιών αναδρομικού τύπου. Αυτές είναι διαδικασίες που καλούν τον εαυτό τους επαναληπτικά ακολουθώντας συγκεκριμένα βήματα μαθηματικών υπολογισμών με διαφορετικές τιμές. Η βάση μίας αναδρομικού τύπου διαδικασίας

είναι ο επαναληπτικός αλγόριθμος που εκτελείται μέχρι να ικανοποιηθεί κάποια συνθήκη. Κάθε υπολογιστικό βήμα προυποθέτει την εκτέλεση ενός απλούστερου υπολογισμού με αναδρομή.

Το παρακάτω πρόγραμμα επιτρέπει την εκτύπωση μιας σειράς χαρακτήρων που τελειώνει με τον χαρακτήρα τελεία '.' κατά την αντίστροφη σειρά απο αυτή που διαβάσθηκε.

```
PROGRAM inverse;
CONST
  point ='.';

PROCEDURE invstr;
VAR
  ch:char;
BEGIN
  READ(ch);
  IF ch <> point THEN invstr; {Αναδρομική κλήση}
  WRITE(ch)
END; {invstr}

BEGIN {main}
  WRITELN('Enter a character string terminated by a period:');
  invstr
END.
```

Φυσικά η κλήση της συνάρτησης γίνεται αναδρομικά.