Το εγχειρίδιο της ΓΛΩΣΣΑΣ

Το παρόν δεν αποσκοπεί στο να διδάξει τη ΓΛΩΣΣΑ, παρά μόνο στο να θυμίζει τη σύνταξη των εντολών στους μαθητές μέσω παραδειγμάτων και στο να πληροφορεί τους καθηγητές για τον τυπικό ορισμό κάθε στοιχείου της ΓΛΩΣΣΑΣ.

Οι τύποι δεδομένων της ΓΛΩΣΣΑΣ

Όταν λέμε τύπο δεδομένων, εννοούμε από ποιο σύνολο τιμών παίρνει τιμές μια έκφραση. Στη ΓΛΩΣΣΑ υπάρχουν τέσσερις τύποι δεδομένων:

- ο ακέραιος, π.χ. -1234567890
- ο πραγματικός, π.χ. 3.14
- οι χαρακτήρες, π.χ. "Ν" ή 'Μυρτώ'
- ο λογικός, δηλαδή ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ

Παρατηρήσεις:

- Αν και προγραμματίζοντας στο χαρτί δεν αντιμετωπίζουμε πρόβλημα μεγέθους στους αριθμούς, στον υπολογιστή έχουμε πάντα περιορισμένο μέγεθος, όσο μεγάλο και αν είναι αυτό. Στην τρέχουσα υλοποίηση του Διερμηνευτή, οι ακέραιοι είναι από -9,223,372,036,854,775,808 μέχρι 9,223,372,036,854,775,807 (64 bit προσημασμένος) και οι πραγματικοί από 5.0 x 10^-324 μέχρι 1.7 x 10^308.
- Πρόβλημα μεγέθους υπάρχει και στους πολύ μικρούς πραγματικούς αριθμούς. Δηλαδή ενώ στο σύνολο των πραγματικών αριθμών δε νοείται ο επόμενος του 1, στον υπολογιστή υπάρχει πάντα ο επόμενος του 1, και στην τρέχουσα υλοποίηση του Διερμηνευτή είναι ο 5.0 x 10^-324. Έτσι αν για παράδειγμα προσπαθήσουμε να προσθέσουμε έναν πολύ μικρό αριθμό στο 1, το άθροισμα θα είναι πάλι 1! Όταν το αποτέλεσμα κάποιας πράξης είναι τόσο μικρός αριθμός που δεν μπορεί να αναπαρασταθεί από τον υπολογιστή, προκύπτει λάθος εκτέλεσης.
- Δεν υπάρχει διαφορετικός τύπος δεδομένων για ένα χαρακτήρα και για πολλούς χαρακτήρες (π.χ. char και string). Η σύγκριση μεταξύ των αλφαριθμητικών γίνεται σύμφωνα με τη διάταξη του προτύπου Unicode, και έτσι τα κεφαλαία είναι διαφορετικά από τα πεζά και τα τονισμένα διαφορετικά από τα άτονα.
- Τα αλφαριθμητικά επιτρέπεται να περικλείονται είτε με απλά είτε με διπλά εισαγωγικά. Για να συμπεριλάβουμε ένα απλό εισαγωγικό σε ένα αλφαριθμητικό πρέπει να γράψουμε δύο απλά (π.χ. 'Γιάννενα πρώτα στ" άρματα') εκτός αν περικλείεται από διπλά εισαγωγικά (π.χ."Γιάννενα πρώτα στ' άρματα"). Το αντίστροφο συμβαίνει για τα αλφαριθμητικά που περικλείονται από διπλά εισαγωγικά.
- Οι πραγματικοί αριθμοί λόγω της περιορισμένης ακρίβειας δε συμπεριφέρονται πάντα όπως θα θέλαμε. Για παράδειγμα αν σε οποιαδήποτε γλώσσα προγραμματισμού προσθέσουμε στο 1 τριάντα φορές το 0.30 δεν θα πάρουμε το 10 αλλά το 10.000000001. Αυτό δημιουργεί προβλήματα σε εντολές επανάληψης, αφού μπορεί να γίνει μία επανάληψη λιγότερη ή περισσότερη. Ο Διερμηνευτής αντιμετωπίζει αυτό το πρόβλημα «περιορίζοντας» την ακρίβεια των πραγματικών σε συγκρίσεις. Για περισσότερες λεπτομέρειες δείτε την περιγραφή της επιλογής «Σημαντικά δεκαδικά ψηφία πραγματικών σε συγκρίσεις» στο αρχείο βοήθειας του Διερμηνευτή.

Οι τελεστές της ΓΛΩΣΣΑΣ

Αριθμητικοί τελεστές

- + (συν): ισχύει για τους ακέραιους και τους πραγματικούς. Προαιρετικά και χωρίς να προτείνεται, μπορείτε να ενεργοποιήσετε τη σχετική επιλογή του Διερμηνευτή ώστε να επιτραπεί η συνένωση χαρακτήρων με τον τελεστή +.
- - (πλην): ισχύει για τους ακέραιους και τους πραγματικούς. Υφίσταται σαν δυαδικός (π.χ. 1 - 2) και σαν μοναδιαίος (π.χ. -1).
- * (επί): ισχύει για τους ακέραιους και τους πραγματικούς.
- / (διά): ισχύει για τους ακέραιους και τους πραγματικούς. Το αποτέλεσμα της διαίρεσης είναι πάντα πραγματικός, ακόμα και αν δεν υπάρχουν δεκαδικά ψηφία (π.χ. 4/2).

- ^ (δύναμη): ισχύει για τους ακέραιους και τους πραγματικούς. Το αποτέλεσμα είναι συνήθως πραγματικός αριθμός, εκτός αν ο εκθέτης είναι θετικός ακέραιος σταθερής αποτίμησης. Για παράδειγμα το 2^2 είναι ακέραιος, ενώ το 2^-2 πραγματικός (= 0.25). Έτσι στην γενική περίπτωση το 2^ι εκλαμβάνεται σαν πραγματικός αριθμός.
- DIV (πηλίκο ακέραιας διαίρεσης): ισχύει μόνο για τους ακέραιους. Για παράδειγμα 7 DIV 2 = 3 (και περισσεύει 1).
- MOD (υπόλοιπο ακέραιας διαίρεσης): ισχύει μόνο για τους ακέραιους. Για παράδειγμα 7 MOD 2 = 1.

Συγκριτικοί τελεστές:

Γενικά οι συγκριτικοί τελεστές εφαρμόζονται μεταξύ των ίδιων τύπων δεδομένων (π.χ. χαρακτήρες >= χαρακτήρες) με εξαίρεση τους ακέραιους και τους πραγματικούς, που μπορεί να συγκρίνονται και μεταξύ τους (π.χ. ακέραιος = πραγματικός). Το αποτέλεσμα των συγκριτικών τελεστών είναι πάντα λογικού τύπου.

- <= ή ≤ (μικρότερο ή ίσο): ισχύει για τους ακέραιους, τους πραγματικούς και τους χαρακτήρες. Οι χαρακτήρες συγκρίνονται σύμφωνα με το πρότυπο Unicode, οπότε A (αγγλικό) < B < a < b < Ά < Α (ελληνικό) < Β < ά < α < β.
- < (μικρότερο): ισχύει για τους ακέραιους, τους πραγματικούς και τους χαρακτήρες. Στις περισσότερες γλώσσες προγραμματισμού οι λογικές τιμές (ΑΛΗΘΗΣ και ΨΕΥΔΗΣ) έχουν διάταξη (ordinal), οπότε υπάρχει ο μεγαλύτερος και ο μικρότερος. Το βιβλίο μαθητή (πιστεύω σωστά) αναφέρει ότι δεν μπορεί να εφαρμοστεί ανισότητα μεταξύ λογικών εκφράσεων.
- = (ίσον): ισχύει για όλους τους τύπους δεδομένων.
- <> ή ≠ (διάφορο): ισχύει για όλους τους τύπους δεδομένων.
- > (μεγαλύτερο): ισχύει για τους ακέραιους, τους πραγματικούς και τους χαρακτήρες.
- >= ή ≥ (μεγαλύτερο ή ίσο): ισχύει για τους ακέραιους, τους πραγματικούς και τους χαρακτήρες.

Λογικοί τελεστές:

Οι λογικοί τελεστές (ΚΑΙ, Η, ΟΧΙ) εφαρμόζονται πάνω σε λογικές εκφράσεις (δηλαδή εκφράσεις που το αποτέλεσμά τους είναι ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ). Το αποτέλεσμά τους είναι πάλι λογικού τύπου δεδομένων.

Στον παρακάτω πίνακα θεωρούμε ότι τα Α και Β είναι λογικές εκφράσεις (π.χ. 1 < 0, β = π, ΑΛΗΘΗΣ).

Α	Β	Α ΚΑΙ Β	Α Η Β	ΟΧΙ Α
ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ
ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	
ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ

ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	

Παρατηρήσεις:

- Υπάρχει ασυμφωνία σχετικά με το αποτέλεσμα των DIV και MOD για αρνητικούς τελεστέους, καθώς και σχετική επιλογή του Διερμηνευτή για να προσαρμόσετε το αποτέλεσμα όπως θέλετε. Όμως, το Φεβρουάριο του 2008 ήρθε διευκρινιστική οδηγία από το Παιδαγωγικό Ινστιτούτο η οποία αναφέρει ότι δεν πρέπει να χρησιμοποιούνται σε ασκήσεις με αρνητικούς αριθμούς οι τελεστές αυτοί.
- Υπάρχει ασάφεια σχετικά με το αν ο τελεστής ύψωσης σε δύναμη ^ εκτελείται από δεξιά προς τα αριστερά ή αντίθετα (προσεταιριστικότητα), καθώς και σχετική επιλογή στο διάλογο ρυθμίσεων του Διερμηνευτή.
- Υπάρχει ασάφεια σχετικά με το αν το ΚΑΙ έχει μεγαλύτερη ή ίση προτεραιότητα με το Η, καθώς και σχετική επιλογή του Διερμηνευτή.
- Πριν την έκδοση 0,89 βήτα οι συγκριτικοί τελεστές μετατρεπόταν αυτόματα στους αντίστοιχους Unicode (≤, ≥, ≠). Από την 0,89 βήτα και μετά υποστηρίζονται μεν οι Unicode, αλλά η προεπιλεγμένη μορφή των τελεστών είναι η ANSI (>=, <=, <>) για συμφωνία με το σχολικό βιβλίο.

Τα μέρη του προγράμματος

Μέρη του προγράμματος λέμε την επικεφαλίδα (δηλαδή το όνομα), το τμήμα δήλωσης σταθερών, το τμήμα δήλωσης μεταβλητών και το κυρίως σώμα του προγράμματος. Στη συνέχεια βέβαια μπορεί να ακολουθούν και υποπρογράμματα. Τα τμήματα δηλώσεων είναι προαιρετικά. Ένα παράδειγμα προγράμματος που χρησιμοποιεί και τα τέσσερα τμήματα είναι το παρακάτω:

```
ΠΡΟΓΡΑΜΜΑ ΕμβαδόνΚύκλου
ΣΤΑΘΕΡΕΣ
  π = 3.14
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: Εμ, Ακ
ΑΡΧΗ
  ΓΡΑΨΕ 'Δώσε την ακτίνα'
  ΔΙΑΒΑΣΕ Ακ
  Εμ <- π*Ακ^2
  ΓΡΑΨΕ 'Το εμβαδόν του κύκλου είναι ', Εμ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατηρήσεις

- Αποδεκτά ονόματα *αναγνωριστικών* (δηλαδή όνομα προγράμματος, σταθεράς ή μεταβλητής) είναι αυτά που ξεκινούν με γράμμα (είτε ελληνικό είτε αγγλικό) και στη συνέχεια περιλαμβάνουν γράμματα, αριθμούς ή την κάτω παύλα (_).

- Στο τμήμα δήλωσης σταθερών επιτρέπεται και η χρήση τελεστών ή των ενσωματωμένων συναρτήσεων, π.χ. οι παρακάτω δηλώσεις θεωρούνται αποδεκτές:
  ΣΤΑΘΕΡΕΣ
    N = 10
    στ = 2 * Τ_Ρ(Ν)
- Στο τμήμα δήλωσης μεταβλητών επιτρέπεται να χρησιμοποιούμε τους τύπους δεδομένων όσες φορές θέλουμε, δηλαδή μπορούμε για παράδειγμα να έχουμε δύο γραμμές με τη λέξη ΠΡΑΓΜΑΤΙΚΕΣ.

Η εντολή ανάθεσης τιμής (<-)

Η εντολή ανάθεσης τιμής μπορεί να μεταφραστεί ως εξής: «Υπολόγισε την τιμή της έκφρασης δεξιά από το σύμβολο ανάθεσης τιμής και βάλε το αποτέλεσμα στη *μεταβλητή* που βρίσκεται αριστερά από την ανάθεση τιμής».

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΗΕντολήΑνάθεσηςΤιμής

!Το πρόγραμμα αυτό δεν περιλαμβάνει «Γράψε». Για να δείτε τις
!τιμές των μεταβλητών κοιτάξτε στην καρτέλα «Μεταβλητές».
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: α
  ΠΡΑΓΜΑΤΙΚΕΣ: π
  ΧΑΡΑΚΤΗΡΕΣ: χ
  ΛΟΓΙΚΕΣ: λ
ΑΡΧΗ
  α <- 1
  π <- 2*α            !Επιτρέπεται κι ας είναι ακέραια η έκφραση
  χ <- 'Κείμενο'

  λ <- α >= π         !Το λ θα γίνει Ψευδής
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατηρήσεις:

- Στο δεξί μέρος μπορεί να βρίσκεται *οτιδήποτε* έχει τιμή, ενώ στο αριστερό μία και μόνο *μεταβλητή* (ή στοιχείο πίνακα) συμβατού τύπου δεδομένων με την έκφραση.
- Η μόνη συμβατή ανάθεση μεταξύ διαφορετικών τύπων δεδομένων είναι από ακέραιο (δεξιά) σε πραγματικό (αριστερά).
- Πριν την έκδοση 0,90 βήτα ο τελεστής ανάθεσης τιμής μετατρεπόταν αυτόματα στη Unicode μορφή του (←). Πλέον υποστηρίζεται και το βελάκι, αλλά η προεπιλεγμένη μορφή είναι η ANSI (<-) για συμφωνία με το σχολικό βιβλίο. Αυτή η συμπεριφορά μπορεί να προσαρμοστεί από το μενού Εργαλεία → Επιλογές του Διερμηνευτή.

Η εντολή ΓΡΑΨΕ

Η εντολή ΓΡΑΨΕ εμφανίζει στην οθόνη τις τιμές μιας λίστας εκφράσεων, χωρισμένων από κόμματα.

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΗΕντολήΓράψε
!Εμφανίζει «10 3.00 κείμενο ΨΕΥΔΗΣ»
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: α
ΑΡΧΗ
  α <- 10
  ΓΡΑΨΕ α, ' ', Τ_Ρ(α - 1), ' κείμενο ', 1 = 3
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατηρήσεις:

- Οι εκφράσεις μπορεί να είναι οποιουδήποτε τύπου δεδομένων. Αν είναι λογικού, στην οθόνη εμφανίζεται ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ.
- Επιτρέπεται εντολή ΓΡΑΨΕ χωρίς παραμέτρους. Απλά κατεβάζει κατά μία γραμμή το δρομέα (σελ. 180 του βιβλίου).
- Για ευκολία σε εισαγωγικές δραστηριότητες των μαθητών, μια εντολή
  ΓΡΑΨΕ 1, 2, 3
  αφήνει κενά μεταξύ των εμφανιζόμενων αριθμών (δηλαδή εμφανίζει 1 2 3 αντί για 123). Αυτή η συμπεριφορά μπορεί να απενεργοποιηθεί από το μενού Εργαλεία → Επιλογές.
- Στη ΓΛΩΣΣΑ δεν υπάρχει εντολή αντίστοιχη της write της Pascal, η οποία να αφήνει το δρομέα στην ίδια γραμμή. Όμως πολλοί συνάδελφοι χρειάστηκαν μία τέτοια εντολή, κυρίως για εμφάνιση δισδιάστατων πινάκων. Άλλοι όμως συνάδελφοι (πολύ σωστά κατά τη γνώμη μου) ανέφεραν ότι δεν πρέπει να εισαχθεί καινούργια εντολή, γιατί έτσι θα παρεκκλίναμε από το βιβλίο.
  Η λύση που υλοποιήθηκε είναι η εξής:
  Αφού αποτιμηθεί όλο το κείμενο που προκύπτει από τη ΓΡΑΨΕ, αν ο τελευταίος χαρακτήρας του είναι το κενό, τότε αυτή μεταφράζεται σε write. Δηλαδή η εντολή
  ΓΡΑΨΕ 'Δώσε το όνομά σου: '
  αφήνει τον δρομέα στην ίδια γραμμή, ενώ η
  ΓΡΑΨΕ 'Δώσε το όνομά σου:'
  αφήνει το δρομέα στην παρακάτω γραμμή.
  Επειδή όμως μερικές φορές χρειάζεται να μην υπάρχει κενό στο τέλος μίας ΓΡΑΨΕ, το τελευταίο κενό **διαγράφεται και δεν εμφανίζεται στην οθόνη εκτέλεσης**. Γι' αυτό το λόγο στο τέλος της πρώτης ΓΡΑΨΕ υπάρχουν δύο κενά, ώστε τελικά να εμφανιστεί το ένα.

Μπορείτε να απενεργοποιήσετε αυτήν τη συμπεριφορά της ΓΡΑΨΕ από το μενού Εργαλεία → Επιλογές.

Η εντολή ΔΙΑΒΑΣΕ

Η εντολή ΔΙΑΒΑΣΕ διαβάζει από το πληκτρολόγιο μία λίστα *μεταβλητών*, χωρισμένων από κόμματα. Για να το επιτύχει αυτό, σε κάθε ΔΙΑΒΑΣΕ η αντίστοιχη γραμμή της καρτέλας «Οθόνη εκτέλεσης» χρωματίζεται για να προσελκύσει την προσοχή του χρήστη και η εκτέλεση του προγράμματος παύει μέχρι να εισαχθούν τα δεδομένα και να πατηθεί το [Enter]. Οι μεταβλητές αυτές μπορεί να είναι οποιουδήποτε τύπου δεδομένων, εκτός από λογικού.

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΗΕντολήΔΙΑΒΑΣΕ
!Διαβάζει μία ακέραια μεταβλητή, μία πραγματική και μία
!αλφαριθμητική. Οι λογικές δεν είναι δυνατόν να διαβαστούν.
!Αντίθετα με τις περισσότερες γλώσσες προγραμματισμού, όταν
!μία ΔΙΑΒΑΣΕ έχει πολλά ορίσματα, αυτά πρέπει να εισαχθούν
!σε ξεχωριστές γραμμές (με [Enter] ανάμεσα από τα ορίσματα).
!Αυτό γίνεται για ευκολία των μαθητών, κοιτάξτε το εγχειρίδιο
!της Γλώσσας για περισσότερες πληροφορίες.
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: α
  ΠΡΑΓΜΑΤΙΚΕΣ: Π[10]
  ΧΑΡΑΚΤΗΡΕΣ: κ
ΑΡΧΗ
  ΔΙΑΒΑΣΕ α, Π[23 DIV 3]          !διαβάζει το α και το Π[7]
  ΔΙΑΒΑΣΕ κ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατηρήσεις

- Δεν επιτρέπεται να διαβαστεί κάτι που δεν είναι μεταβλητή. Για παράδειγμα, η εντολή
  ΔΙΑΒΑΣΕ α + β, -π, 24, 'κείμενο'
  είναι τελείως λάθος.
- Δεν είναι δυνατό να διαβαστούν λογικές μεταβλητές. Αντί αυτού μπορούμε να διαβάσουμε έναν ακέραιο ή ένα χαρακτήρα και στη συνέχεια να δώσουμε την αντίστοιχη τιμή στη λογική μεταβλητή με μια εντολή Αν, όπως στο πιο κάτω παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ  ΔιάβασμαΛογικήςΜεταβλητής
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: ι
  ΛΟΓΙΚΕΣ: φύλο
```

```
ΑΡΧΗ
  ΓΡΑΨΕ 'Δώσε 0 αν είσαι γυναίκα και 1 αν είσαι άντρας: '
  ΔΙΑΒΑΣΕ ι
  ΑΝ ι = 0 ΤΟΤΕ
    φύλο <- ΑΛΗΘΗΣ
  ΑΛΛΙΩΣ
    φύλο <- ΨΕΥΔΗΣ
  ΤΕΛΟΣ_ΑΝ
  !Η παραπάνω εντολή Αν ισοδυναμεί με την εντολή «φύλο <- ι = 0»
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

- Στο Διερμηνευτή υπάρχει η επιλογή να ελέγχονται τα δεδομένα κατά την είσοδο. Αυτό σημαίνει ότι αν το πρόγραμμα προσπαθεί να διαβάσει έναν ακέραιο, η περιοχή εισόδου δεν επιτρέπει να πατηθούν γράμματα παρά μόνο νούμερα. Αντίστοιχα στους πραγματικούς επιτρέπει και μία τελεία.
- Σε εντολές του τύπου ΔΙΑΒΑΣΕ χ, ψ, ω δεν επιτρέπεται να εισαχθούν όλα τα δεδομένα σε μία γραμμή, αλλά πρέπει κάθε τιμή να εισάγεται χωριστά πατώντας ισάριθμες φορές το [Enter] (αντίθετα από τις περισσότερες γλώσσες προγραμματισμού). Αυτό γίνεται για ευκολία των μαθητών, και για να μη δημιουργηθούν προβλήματα: για παράδειγμα η παράλληλη χρήση των read και readln της Pascal μπερδεύει εύκολα ακόμα και πεπειραμένους προγραμματιστές, ενώ στην C η scanf δε μπορεί να διαβάσει αλφαριθμητικά που να περιέχουν κενά.
  Για του λόγου το αληθές: σε μια εντολή ΔΙΑΒΑΣΕ όνομα, ηλικία αν ο χρήστης έδινε «Άλκης Γεωργόπουλος 35» πού θα έπρεπε να σταματήσει το διάβασμα του αλφαριθμητικού;

Η εντολή ΑΝ

Η εντολή ΑΝ χρησιμοποιείται όταν χρειάζεται διακλάδωση της ροής του προγράμματος, ανάλογα με κάποια συνθήκη.

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΗΕντολήΑν
!Διαβάζει ένα βαθμό και εμφανίζει
!στο χρήστη αν πέρασε ή αν κόπηκε.
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: βαθμός
ΑΡΧΗ
  ΓΡΑΨΕ 'Δώσε βαθμό:  '
  ΔΙΑΒΑΣΕ βαθμός

  ΑΝ βαθμός < 0 ΤΟΤΕ
    ΓΡΑΨΕ 'Δεν υπάρχει αρνητικός βαθμός'
```

```
  ΑΛΛΙΩΣ_ΑΝ βαθμός < 10 ΤΟΤΕ
    ΓΡΑΨΕ 'Κόπηκες'
  ΑΛΛΙΩΣ_ΑΝ βαθμός <= 20 ΤΟΤΕ
    ΓΡΑΨΕ 'Πέρασες'
  ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'Ο βαθμός πρέπει να είναι στην εικοσαβάθμια,',
    & ' όχι στην εκατοσταβάθμια κλίμακα'
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατήρηση:

Η εντολή ΑΛΛΙΩΣ_ΑΝ είναι διαφορετική από την ΑΛΛΙΩΣ ΑΝ. Η δεύτερη είναι δύο εντολές, δηλαδή το μέρος ΑΛΛΙΩΣ μιας εντολής ΑΝ και μία δεύτερη ΑΝ στη συνέχεια (η οποία μάλιστα δεν επιτρέπεται να βρίσκεται στην ίδια γραμμή, θα πρέπει να γραφεί στην παρακάτω). Έτσι στην ΑΛΛΙΩΣ_ΑΝ χρειάζεται ένα ΤΕΛΟΣ_ΑΝ, ενώ στην ΑΛΛΙΩΣ ΑΝ χρειάζονται όσα και τα ΑΝ.

Η εντολή ΕΠΙΛΕΞΕ

Η εντολή ΕΠΙΛΕΞΕ χρησιμοποιείται αντί για πολλές Αν ... αλλιώς_αν ... όταν θέλουμε να πάρουμε περιπτώσεις ανάλογα με την τιμή *μίας μόνο έκφρασης*. Δηλαδή δεν μπορεί να αντικαταστήσει πολλές Αν οι οποίες αναφέρονται σε διαφορετικές συνθήκες.

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ  ΗΕντολήΕΠΙΛΕΞΕ
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: βαθμός
ΑΡΧΗ
  ΓΡΑΨΕ 'Δώσε βαθμό:  '
  ΔΙΑΒΑΣΕ βαθμός
  ΕΠΙΛΕΞΕ βαθμός
    ΠΕΡΙΠΤΩΣΗ 15..20                    !15 <= βαθμός <= 20
      ΓΡΑΨΕ 'Είσαι άξιος'
    ΠΕΡΙΠΤΩΣΗ 10..15                    !10 <= βαθμός < 15
      ΓΡΑΨΕ 'Καλούτσικα πήγες'
    ΠΕΡΙΠΤΩΣΗ 0                         !βαθμός = 0
      ΓΡΑΨΕ 'Δεν προσήλθες στις εξετάσεις'
    ΠΕΡΙΠΤΩΣΗ 0..10                     !0 < βαθμός < 10
      ΓΡΑΨΕ 'Κόπηκες'
    ΠΕΡΙΠΤΩΣΗ > 20
      ΓΡΑΨΕ 'Ο βαθμός θα πρέπει να είναι στην εικοσαβάθμια, ',
```

```
    & 'όχι στην εκατοσταβάθμια κλίμακα'
  ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ
    ΓΡΑΨΕ 'Έδωσες άκυρο βαθμό'
 ΤΕΛΟΣ_ΕΠΙΛΟΓΩΝ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατηρήσεις:

- Η έκφραση στην ΕΠΙΛΕΞΕ μπορεί να είναι οποιουδήποτε τύπου δεδομένων. Παραδείγματα από το σχολικό εγχειρίδιο: με πραγματικούς αριθμούς στη σελίδα 75 του τετραδίου μαθητή, με αλφαριθμητικά σελίδα στη 269 του βιβλίου μαθητή.
- Επιτρέπεται η χρήση συγκριτικών τελεστών στις περιπτώσεις της ΕΠΙΛΕΞΕ, για παράδειγμα ΠΕΡΙΠΤΩΣΗ > 20. Σχετικό παράδειγμα υπάρχει στη σελίδα 75 του τετραδίου μαθητή.
- Όμως, **δεν** επιτρέπεται η χρήση λογικών τελεστών, δηλαδή για παράδειγμα η ΠΕΡΙΠΤΩΣΗ > 20 ΚΑΙ < 40 δεν είναι αποδεκτή. Για περισσότερες πληροφορίες επισκεφτείτε το σχετικό θέμα στο Στέκι των Πληροφορικών.
- Προσέξτε τη σειρά των περιπτώσεων. Ο βαθμός θεωρείται πραγματικός και επομένως μια τιμή = 9.5 θεωρείται αποδεκτή και έπρεπε να εμφανίσει το μήνυμα 'Κόπηκες'. Αν όμως η περίπτωση 0 .. 10 προηγούταν της 10 .. 15 τότε το 10 θα πήγαινε στην πρώτη περίπτωση και το αποτέλεσμα θα ήταν πάλι 'Κόπηκες'. *Δηλαδή στην ΕΠΙΛΕΞΕ, όταν έχουμε να κάνουμε με πραγματικούς, πρέπει να προηγούνται τα κλειστά διαστήματα, όπως το [*10 .. 15*) και μετά τα ανοιχτά, όπως το (*0 .. 10*).*
- Η κάθε ΠΕΡΙΠΤΩΣΗ μπορεί να περιέχει πολλές εκφράσεις χωρισμένες με κόμματα, π.χ.

  ```
  ΠΕΡΙΠΤΩΣΗ Τ_Ρ(Ν/2) .. α, β + 25, 10 .. 9
  ```

  Φυσικά το διάστημα 10 .. 9 είναι σαν να μην υπάρχει στη συγκεκριμένη περίπτωση, αφού το κάτω άκρο είναι μεγαλύτερο από το πάνω άκρο. Η έκφραση β + 25 δεν είναι διάστημα, αλλά συγκεκριμένη τιμή. Η παραπάνω λοιπόν περίπτωση ισοδυναμεί με την εντολή:

  ```
  ΑΝ (επιλογέας >= Τ_Ρ(Ν/2) ΚΑΙ επιλογέας <= α) Η
     (επιλογέας = β + 25) Η
     (επιλογέας >= 10 ΚΑΙ επιλογέας <= 9) ΤΟΤΕ ...
  ```

- Από το Διερμηνευτή της Γλώσσας θεωρείται αποδεκτή μία ΕΠΙΛΕΞΕ που περιέχει μόνο ΠΕΡΙΠΤΩΣΗ ΑΛΛΙΩΣ , αλλά δεν είναι αποδεκτή μία ΕΠΙΛΕΞΕ χωρίς καμία ΠΕΡΙΠΤΩΣΗ.
- Η έκφραση στην ΕΠΙΛΕΞΕ αποτιμάται μία φορά, όχι διαδοχικά σε κάθε περίπτωση. Δηλαδή αν η έκφραση αυτή περιέχει και κλήση σε συνάρτηση, η συνάρτηση θα καλεστεί μόνο κατά την ΕΠΙΛΕΞΕ και όχι σε κάθε ΠΕΡΙΠΤΩΣΗ.

Η εντολή ΓΙΑ

Η εντολή ΓΙΑ είναι η πιο απλή εντολή επανάληψης. Χρησιμοποιεί ένα μετρητή για να μετράει πόσες επαναλήψεις γίνονται. Επιλέγουμε να τη χρησιμοποιήσουμε όταν γνωρίζουμε από πριν πόσες επαναλήψεις θέλουμε.

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΗΕντολήΓια
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: π
ΑΡΧΗ
  ΓΙΑ π ΑΠΟ 10 ΜΕΧΡΙ 1 ΜΕ_ΒΗΜΑ -0.5
    ΓΡΑΨΕ π
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατηρήσεις:

- Στο βιβλίο μαθητή το μέρος ΜΕ ΒΗΜΑ της εντολής ΓΙΑ αναφέρεται τέσσερις φορές χωρίς κάτω παύλα, ενώ σε όλες τις υπόλοιπες αναφορές των σχολικών εγχειριδίων είναι με κάτω παύλα. Έτσι ο Διερμηνευτής τα θεωρεί και τα δύο αποδεκτά εκτός εάν αλλάξετε τη σχετική επιλογή από το μενού Εργαλεία.
- Αν το βήμα παραληφθεί, θεωρείται ίσο με 1.
- Στα μέρη ΑΠΟ, ΜΕΧΡΙ και ΜΕ_ΒΗΜΑ μπορούν να χρησιμοποιηθούν ακέραιες ή πραγματικές εκφράσεις. Βέβαια αν ο μετρητής είναι ακέραιος, δεν επιτρέπεται πραγματική τιμή στο ΑΠΟ ή στο ΜΕ_ΒΗΜΑ. Οι εκφράσεις αυτές αποτιμούνται μία μόνο φορά, δηλαδή αν περιέχουν κλήση συνάρτησης αυτή θα καλεστεί μία φορά.
- Η ΓΙΑ είναι η εντολή με τις πιο μεγάλες διαφορές στην υλοποίησή της στις γλώσσες προγραμματισμού Basic και Pascal (τη C τη βγάζω απ' έξω γιατί η for της δεν αντιστοιχεί ακριβώς με την έννοια του μετρητή, της αρχικής τιμής και της τιμής τέλους). Δηλαδή:
  - Στην Pascal δεν επιτρέπεται βήμα, παρά μόνο το -1 όταν κάνουμε downto αντί για to.
  - Μετά από ένα «for i := 1 to 10» στην Pascal το i είναι 10, ενώ στην Basic και στο Delphi είναι 11. Το τελευταίο ισχύει και στο Διερμηνευτή, για συμφωνία με το σχολικό βιβλίο.
  - Στο Delphi και στην Pascal αν δεν πρόκειται να γίνει καμιά επανάληψη (π.χ. for i := 1 to 0), η αρχική τιμή δεν ανατίθεται στο μετρητή. Στην Basic όπως και στο Διερμηνευτή ο μετρητής έτσι κι αλλιώς παίρνει την αρχική τιμή.

  Ουσιαστικά η μετάφραση του ΓΙΑ με χρήση της ΟΣΟ είναι:

```
μετρητής <- έκφραση_αρχής
τιμή_τέλους <- έκφραση_τέλους        !ΓΙΑ να μη γίνεται πολλές
```

```
τιμή_βήματος <- έκφραση_βήματος            !φορές η αποτίμηση
ΟΣΟ (μετρητής (>= Η <=) έκφραση_τέλους) ΕΠΑΝΑΛΑΒΕ
  εντολές
  μετρητής <- μετρητής + τιμή_βήματος
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Οι τιμή_τέλους, τιμή_βήματος είναι βοηθητικές μεταβλητές. Το (>= Η <=) εξαρτάται από την τιμή_βήματος. Αν είναι θετική, χρησιμοποιούμε το μικρότερο ή ίσο, αλλιώς το μεγαλύτερο ή ίσο.

Η εντολή ΟΣΟ

Η εντολή ΟΣΟ είναι η πιο ισχυρή εντολή επανάληψης. Χρησιμοποιείται συνήθως όταν δεν ξέρουμε τον αριθμό των επαναλήψεων, αλλά οι επαναλήψεις εξαρτώνται από κάποια συνθήκη. Την προτιμούμε από την ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ όταν είναι πιθανό να μη γίνει καμία επανάληψη, γιατί στην ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ γίνεται πάντα τουλάχιστον μία.

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΗΕντολήΌσο
!Μετράει (με χαζό τρόπο) πόσες φορές μπορεί
!να διαιρεθεί διαδοχικά ένας αριθμός με το 2.
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: ι, α
ΑΡΧΗ
  ΔΙΑΒΑΣΕ α
  ι <- 0
  ΟΣΟ α >= 2 ΕΠΑΝΑΛΑΒΕ
    α <- α DIV 2
    ι <- ι + 1
  ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
  ΓΡΑΨΕ 'Ο αριθμός μπορεί να διαιρεθεί ', ι, ' φορές με το 2.'
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατήρηση

Η ΟΣΟ δεν περιλαμβάνει μετρητή όπως η ΓΙΑ. Αν θέλετε να βάλετε μετρητή στην ΟΣΟ, θυμηθείτε τα τρία βήματα του μετρητή:

1. Αρχικοποίηση πριν την ΟΣΟ (π.χ. ι <- 1)
2. Μεταβολή πριν το ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ (π.χ. ι <- ι + 1)
3. Συνθήκη τέλους στην ΟΣΟ (π.χ. ΟΣΟ ι <= 10)

Η εντολή ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ

Η εντολή ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ χρησιμοποιείται όταν δεν υπάρχει συγκεκριμένος αριθμός επαναλήψεων, αλλά οι επαναλήψεις εξαρτώνται από κάποια συνθήκη. Την προτιμούμε από την ΟΣΟ όταν θέλουμε να γίνει τουλάχιστον μία επανάληψη. Είναι η αντίστοιχη της repeat - until της Pascal και της do - while στην Basic και στη C.

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΗΕντολήΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
!Διαβάζει έναν ακέραιο υποχρεώνοντας το
!χρήστη να είναι μεταξύ 0 και 20.
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: βαθμός
ΑΡΧΗ
  ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ 'Δώσε βαθμό:  '
    ΔΙΑΒΑΣΕ βαθμός

  ΜΕΧΡΙΣ_ΟΤΟΥ βαθμός >= 0 ΚΑΙ βαθμός <= 20
  ΓΡΑΨΕ "Εδώσες ", βαθμός
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατήρηση

Η ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ δεν περιλαμβάνει μετρητή όπως η ΓΙΑ. Αν θέλετε να βάλετε μετρητή στην ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ, θυμηθείτε τα τρία βήματα του μετρητή:

1. Αρχικοποίηση πριν την ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ (π.χ. ι <- 1)
2. Μεταβολή πριν το ΜΕΧΡΙΣ_ΟΤΟΥ (π.χ. ι <- ι + 1)
3. Συνθήκη τέλους στη ΜΕΧΡΙΣ_ΟΤΟΥ (π.χ. ΜΕΧΡΙΣ_ΟΤΟΥ ι = 10)

Ενσωματωμένες συναρτήσεις

Η ΓΛΩΣΣΑ έχει ενσωματωμένες τις εξής συναρτήσεις:

- Α_Μ(πραγματικός): το Ακέραιο Μέρος ενός πραγματικού. Δέχεται πραγματικό και επιστρέφει ακέραιο (όπου μπορεί να μπει πραγματικός μπορεί και ακέραιος, λόγω του ότι οι ακέραιοι είναι υποσύνολο των πραγματικών. Το ίδιο ισχύει και στην ανάθεση τιμής π <- 1 όπου το π είναι πραγματικός και το 1 ακέραιος).
- Α_Τ(αριθμός): η Απόλυτη Τιμή ενός αριθμού. Δέχεται ακέραιο και επιστρέφει ακέραιο ή δέχεται πραγματικό και επιστρέφει πραγματικό.
- Ε(πραγματικός): η Εκθετική συνάρτηση (e^x).
- ΕΦ(πραγματικός): η ΕΦαπτομένη ενός πραγματικού. Επιστρέφει πραγματικό.
- ΗΜ(πραγματικός): το ΗΜίτονο ενός πραγματικού. Επιστρέφει πραγματικό.

- ΛΟΓ(πραγματικός): ο φυσικός ΛΟΓάριθμος ενός πραγματικού. Επιστρέφει πραγματικό.
- ΣΥΝ(πραγματικός): το ΣΥΝημίτονο ενός πραγματικού. Επιστρέφει πραγματικό.
- Τ_Ρ(πραγματικός): η Τετραγωνική Ρίζα ενός πραγματικού. Ισοδυναμεί με x^(1/2).

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΕνσωματωμένεςΣυναρτήσεις
!Επιδεικνύει τις ενσωματωμένες συναρτήσεις της Γλώσσας
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: π
ΑΡΧΗ
  π <- -3.2
  ΓΡΑΨΕ 'Ακέραιο μέρος του ', π, ': ', Α_Μ(π)
  ΓΡΑΨΕ 'Απόλυτη τιμή του ', π, ': ', Α_Τ(π)
  ΓΡΑΨΕ 'Εκθετική συνάρτηση του ', π, ': ', Ε(π)
  ΓΡΑΨΕ 'Εφαπτομένη του ', π, ': ', ΕΦ(π)
  ΓΡΑΨΕ 'Ημίτονο του ', π, ': ', ΗΜ(π)
!Στο δεκαδικό λογάριθμο δεν επιτρέπονται αρνητικοί αριθμοί
  ΓΡΑΨΕ 'Δεκαδικός λογάριθμος του ', Α_Τ(π), ': ', ΛΟΓ(Α_Τ(π))
  ΓΡΑΨΕ 'Συνημίτονο του ', π, ': ', ΣΥΝ(π)
!Στην τετραγωνική ρίζα δεν επιτρέπονται αρνητικοί αριθμοί
  ΓΡΑΨΕ 'Τετραγωνική ρίζα του ', Α_Τ(π), ': ', Τ_Ρ(Α_Τ(π))
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Παρατηρήσεις:

- Η παράμετρος των ΕΦ, ΗΜ και ΣΥΝ είναι σε μοίρες, όχι σε ακτίνια (σελίδα 62 του τετραδίου μαθητή). Εάν θέλετε μπορείτε να αλλάξετε αυτή τη συμπεριφορά από τις επιλογές του Διερμηνευτή, αλλά δεν προτείνεται.
- Υπάρχει ασάφεια σχετικά με το αν η Α_Τ(-5.5) κάνει -5 ή -6, δείτε το σχετικό θέμα στο Στέκι των Πληροφορικών. Μπορείτε να ορίσετε τη συμπεριφορά που θέλετε από τις επιλογές του Διερμηνευτή.
- Οι ενσωματωμένες συναρτήσεις είναι σταθερής αποτίμησης, δηλαδή μπορούν να χρησιμοποιηθούν σε δηλώσεις σταθερών, σε όρια πινάκων, σε εντολές ΠΕΡΙΠΤΩΣΗ κ.α.

Η εντολή ΚΑΛΕΣΕ

Η εντολή ΚΑΛΕΣΕ μας επιτρέπει να καλέσουμε μία διαδικασία (όχι όμως συναρτήσεις, αυτές καλούνται μέσα από εκφράσεις).

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΗΕντολήΚάλεσε
!Επιδεικνύει απλά τη χρήση της εντολής.
ΑΡΧΗ
  ΚΑΛΕΣΕ ΓράψεΌνομα('My name is Bond. James Bond.')
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ ΓράψεΌνομα(όνομα)
ΜΕΤΑΒΛΗΤΕΣ
  ΧΑΡΑΚΤΗΡΕΣ: όνομα
ΑΡΧΗ
  ΓΡΑΨΕ όνομα
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Παρατηρήσεις:

- Ο αριθμός και ο τύπος των παραμέτρων καθορίζονται από την αντίστοιχη διαδικασία.

  Κοιτάξτε τις ιστοσελίδες για τις διαδικασίες και τη μεταβίβαση παραμέτρων για περισσότερες πληροφορίες.
- Αν μία διαδικασία δεν έχει παραμέτρους, δεν πρέπει να μπουν παρενθέσεις κατά την κλήση της.

Διαδικασίες

Οι διαδικασίες είναι υποπρογράμματα που γράφονται μετά το κυρίως πρόγραμμα. Μπορούν να κληθούν με την εντολή ΚΑΔΕΣΕ από οποιοδήποτε σημείο του προγράμματος. Οι παράμετροι αναφέρονται ονομαστικά κατά τη δήλωση της διαδικασίας και στη συνέχεια ο τύπος τους δηλώνεται στο τμήμα «Μεταβλητές».

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ ΜίαΔιαδικασία
!Δημιουργία διαδικασίας και κλήση της με την Κάλεσε.
ΑΡΧΗ
  ΚΑΛΕΣΕ ΓράψεΌνομα('My name is Bond. James Bond.')
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ ΓράψεΌνομα(όνομα)
ΜΕΤΑΒΛΗΤΕΣ
  ΧΑΡΑΚΤΗΡΕΣ: όνομα
ΑΡΧΗ
```

```
  ΓΡΑΨΕ όνομα
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ
```

Παρατηρήσεις:

- Σαν παράμετρος μπορεί να περαστεί οτιδήποτε έχει τιμή, όπως μεταβλητές, πίνακες, σταθερές, εκφράσεις, ακόμα και εκφράσεις που περιέχουν κλήση συναρτήσεων.
- Αν μία διαδικασία κληθεί με μεταβλητή σαν παράμετρο και αλλαχθεί η τιμή της μέσα στην διαδικασία, τότε η τιμή της μεταβλητής στο καλών υποπρόγραμμα θα ενημερωθεί (αλλαχθεί) μετά την εκτέλεση της εντολής ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ.
- Μία διαδικασία χωρίς παραμέτρους δηλώνεται και καλείται χωρίς παρενθέσεις.
- Επιτρέπονται μέχρι 1000 *αναδρομικές* κλήσεις (μία διαδικασία να καλεί τον εαυτό της). Αν γίνουν παραπάνω εμφανίζεται το μήνυμα «Υπερχείλιση στοίβας» και το πρόγραμμα τερματίζεται. Φυσικά *επαναληπτικές* κλήσεις επιτρέπονται απεριόριστες.
- Μπορείτε να δείτε κάποιες ιδιαιτερότητες της ΓΛΩΣΣΑΣ στη μεταβίβαση παραμέτρων.

Συναρτήσεις

Η ΓΛΩΣΣΑ επιτρέπει τη δημιουργία συναρτήσεων από το χρήστη, οι οποίες μπορούν στη συνέχεια να χρησιμοποιηθούν όπως και οι ενσωματωμένες συναρτήσεις. Το αποτέλεσμα κάθε συνάρτησης έχει πάντα τιμή κάποιου τύπου δεδομένων και δηλώνεται μετά τις παραμέτρους της συνάρτησης (ΑΚΕΡΑΙΑ, ΠΡΑΓΜΑΤΙΚΗ, ΧΑΡΑΚΤΗΡΑΣ ή ΛΟΓΙΚΗ). Οι συναρτήσεις δεν επιτρέπεται να επιστρέφουν πίνακες. Όλες οι συναρτήσεις πρέπει να περιέχουν μία εντολή του τύπου ΌνομαΣυνάρτησης <- τιμή, ώστε να επιστρέφουν κάποια τιμή στο καλών υποπρόγραμμα.

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ Συναρτήσεις
!Παράδειγμα υλοποίησης συναρτήσεων.
!Υλοποιεί τις συναρτήσεις ΑπόλυτηΤιμή και Μέγιστος.
ΑΡΧΗ
  ΓΡΑΨΕ Μέγιστος(ΑπόλυτηΤιμή(-2), 1)            !Θα γράψει «2»
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΣΥΝΑΡΤΗΣΗ ΑπόλυτηΤιμή(χ): ΑΚΕΡΑΙΑ
!Βρίσκει την απόλυτη τιμή του χ χωρίς να χρησιμοποιήσει την Α_Τ
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: χ
ΑΡΧΗ
  ΑΝ χ >= 0 ΤΟΤΕ
    ΑπόλυτηΤιμή <- χ
  ΑΛΛΙΩΣ
```

```
    ΑπόλυτηΤιμή <- -χ
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ

ΣΥΝΑΡΤΗΣΗ Μέγιστος(α, β): ΑΚΕΡΑΙΑ
!Επιστρέφει το Μέγιστο μεταξύ των α και β
ΜΕΤΑΒΛΗΤΕΣ
  ΑΚΕΡΑΙΕΣ: α, β
ΑΡΧΗ
  ΑΝ α >= β ΤΟΤΕ
    Μέγιστος <- α
  ΑΛΛΙΩΣ
    Μέγιστος <- β
  ΤΕΛΟΣ_ΑΝ
ΤΕΛΟΣ_ΣΥΝΑΡΤΗΣΗΣ
```

Παρατηρήσεις:

- Αν δεν υπάρχει εντολή τύπου ΌνομαΣυνάρτησης <- τιμή στο εσωτερικό μιας συνάρτησης ο Διερμηνευτής εμφανίζει σφάλμα εκτέλεσης, ώστε να ειδοποιηθεί ο χρήστης για την παράλειψη.
- Αν το όνομα μιας συνάρτησης εμφανιστεί οπουδήποτε αλλού εκτός από το αριστερό μέρος μιας ανάθεσης τιμής προκαλείται *αναδρομή*. Το ίδιο συμβαίνει στις περισσότερες γλώσσες προγραμματισμού.
- Σαν παράμετρος μπορεί να περαστεί οτιδήποτε έχει τιμή, όπως μεταβλητές, πίνακες, σταθερές, εκφράσεις, ακόμα και εκφράσεις που περιέχουν κλήση συναρτήσεων.
- Οι παράμετροι στις συναρτήσεις περνιούνται με *τιμή*, δηλαδή οποιαδήποτε αλλαγή στις τυπικές παραμέτρους μιας συνάρτησης δεν επηρεάζει τις πραγματικές παραμέτρους της συνάρτησης.
- Μία συνάρτηση χωρίς παραμέτρους δηλώνεται και καλείται χωρίς παρενθέσεις.
- Επιτρέπονται μέχρι 1000 *αναδρομικές* κλήσεις (μία συνάρτηση να καλεί τον εαυτό της). Αν γίνουν παραπάνω εμφανίζεται το μήνυμα «Υπερχείλιση στοίβας» και το πρόγραμμα τερματίζεται. Φυσικά *επαναληπτικές* κλήσεις επιτρέπονται απεριόριστες.
- Μπορείτε να δείτε κάποιες ιδιαιτερότητες της ΓΛΩΣΣΑΣ στη μεταβίβαση παραμέτρων.

Μεταβίβαση παραμέτρων σε υποπρογράμματα

Η ΓΛΩΣΣΑ μοιάζει λίγο με την Basic στη μεταβίβαση παραμέτρων, αλλά είναι πάρα πολύ διαφορετική από την Pascal. Η Basic ακολουθεί το μηχανισμό μεταβίβασης παραμέτρων *με αναφορά*, *όποτε* αυτό είναι δυνατό. Ας θεωρήσουμε το παρακάτω παράδειγμα:

Παράδειγμα:

```
ΠΡΟΓΡΑΜΜΑ  ΜεταβίβασηΠαραμέτρων
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: π
  ΑΚΕΡΑΙΕΣ: α
ΑΡΧΗ
  π <- 1
  ΓΡΑΨΕ 'π = ', π                          !Γράφει 1
  ΚΑΛΕΣΕ Δ(π)        !Καλείται με μεταβλητή, οπότε το π αλλάζει
  ΓΡΑΨΕ 'π = ', π                          !Γράφει 2
  α <- 1
  ΓΡΑΨΕ 'α = ', α                          !Γράφει 1
! Κάλεσε Δ(α)    !Αυτό δεν επιτρέπεται, προσπαθούμε να περάσουμε
                !με αναφορά ακέραιο ενώ απαιτείται πραγματικός.
  ΚΑΛΕΣΕ Δ((α))  !Βάζοντας το α σε παρένθεση το κάνουμε έκφραση,
!οπότε περνιέται με τιμή. Το ίδιο θα γινόταν με Κάλεσε Δ(α + 0).
  ΓΡΑΨΕ 'α = ', α
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ

ΔΙΑΔΙΚΑΣΙΑ Δ(β)
!Αυξάνει το β. Αν περαστεί μεταβλητή (= με αναφορά) αλλάζει και
!η μεταβλητή του κυρίως προγράμματος. Αν περαστεί έκφραση (= με
!τιμή) το β αλλάζει μόνο τοπικά.
ΜΕΤΑΒΛΗΤΕΣ
  ΠΡΑΓΜΑΤΙΚΕΣ: β
ΑΡΧΗ
  β <- β + 1
ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Δ
```

Παρατηρήσεις:

- Ο παραπάνω τρόπος υλοποίησης ήταν απαραίτητος ώστε να είναι αποδεκτά *όλα* τα παραδείγματα του σχολικού βιβλίου (δείτε τα παραδείγματα του βιβλίου Πύργοι του Ανόι και Αναδρομικό παραγοντικό, περιλαμβάνονται στα παραδείγματα του Διερμηνευτή).
- Αν καλέσουμε με παράμετρο *μεταβλητή* κάποια διαδικασία (αυτό δεν ισχύει για τις συναρτήσεις) και αυτή αλλάξει την τιμή της, η νέα τιμή θα επιστραφεί στο κυρίως πρόγραμμα. Η τυπική και η ουσιαστική παράμετρος θα πρέπει να είναι του ίδιου τύπου δεδομένων, δεν επιτρέπεται καν να περάσουμε ακέραιο σε κάποια διαδικασία που περιμένει πραγματικό αριθμό.

- Συνεπάγεται ότι σε όλα τα υποπρογράμματα θα πρέπει να προσέχουμε πότε αλλάζουμε τις παραμέτρους γιατί οι αλλαγές μπορεί να επηρεάσουν και τις τιμές των ουσιαστικών παραμέτρων.
- Αν καλέσουμε με παράμετρο *έκφραση* κάποιο υποπρόγραμμα (είτε διαδικασία είτε συνάρτηση) και αυτό αλλάξει την τιμή της, η νέα τιμή φυσικά δεν μπορεί να επιστραφεί στο κυρίως πρόγραμμα. Η έκφραση πρέπει πάλι να είναι του ίδιου τύπου με την τυπική παράμετρο, με τη διαφορά ότι επιτρέπεται πέρασμα ακεραίου σε υποπρόγραμμα που περιμένει πραγματικό αριθμό (όπως και στην ανάθεση τιμής).
- Αν θέλουμε να περάσουμε *με τιμή* κάποια μεταβλητή, τότε είμαστε αναγκασμένοι να κάνουμε κάποια «πράξη» με αυτήν ώστε να πάψει να είναι μεταβλητή (π.χ. 1*χ, χ + 0). Η πιο απλή «πράξη» που μπορούμε να κάνουμε είναι να τη βάλουμε μέσα σε παρένθεση. Η ίδια τακτική εφαρμόζεται και στην Basic.
- Οι ομοιότητες με την Basic σταματούν εδώ. Το σχολικό βιβλίο στη σελίδα 218 του βιβλίου μαθητή περιγράφει μηχανισμό μεταβίβασης παραμέτρων *με αντιγραφή (copy in - copy out)*, όχι με αναφορά. Αυτό ακριβώς έχει υλοποιηθεί και στο Διερμηνευτή. Οι δύο αυτοί μηχανισμοί μοιάζουν και τις περισσότερες φορές συμπεριφέρονται με τον ίδιο τρόπο, αλλά υπάρχουν τρία λεπτά σημεία όπου είναι τελείως διαφορετικοί:
    - Όταν μέσα σε μία διαδικασία αλλάζει η τιμή μιας παραμέτρου με αναφορά, ταυτόχρονα γίνεται η αλλαγή στην αντίστοιχη μεταβλητή του κυρίως προγράμματος. Αυτό συμβαίνει επειδή στην πράξη δεν υπάρχουν δύο μεταβλητές, απλά ορίζουμε δύο διαφορετικά ονόματα για την ίδια μεταβλητή. Αντίθετα στο μηχανισμό copy in - copy out υπάρχουν *δύο* μεταβλητές και η δεύτερη (της διαδικασίας) αντιγράφεται στην πρώτη με την εντολή ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ. Μπορείτε να το δοκιμάσετε στο Διερμηνευτή: σταματήστε την εκτέλεση του προγράμματος ΜεταβίβασηΠαραμέτρων αφού ξεκινήσει η ΚΑΛΕΣΕ Δ(π) αλλά *πριν* εκτελεστεί το ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Δ. Το β θα έχει γίνει 2 αλλά αν από το πλαίσιο «Κληθέντα υποπρογράμματα» επιλέξουμε να παρακολουθήσουμε το κυρίως πρόγραμμα, θα δούμε ότι το π έχει ακόμα την τιμή 1. Γίνεται 2 ακριβώς μετά το ΤΕΛΟΣ_ΔΙΑΔΙΚΑΣΙΑΣ Δ.
- Η δημιουργία διαφορετικών μεταβλητών για κάθε κλήση υποπρογράμματος σπαταλά γρήγορα τη διαθέσιμη μνήμη. Ας υποθέσουμε ότι θέλουμε να ταξινομήσουμε έναν πίνακα 5000 ονομάτων με τον αναδρομικό αλγόριθμο QuickSort. Ένας τέτοιος πίνακας μπορεί να χρειαστεί 1 Mb μνήμης. Αν γίνουν 20 αναδρομικές κλήσεις, τότε χρειαζόμαστε 20 Mb RAM στον υπολογιστή μας *μόνο* για τον πίνακα, χωρίς να υπολογίσουμε τη μνήμη που απαιτούν τα Windows και ο Διερμηνευτής. Ευτυχώς στα εκπαιδευτικά προγράμματα που καλούμαστε να υλοποιήσουμε δε συνηθίζονται τόσο μεγάλα μεγέθη πινάκων και αναδρομής.
- Οι δύο παραπάνω διαφορές του μηχανισμού copy in - copy out από αυτόν με αναφορά δεν ήταν ιδιαίτερα «σημαντικές» επειδή το αποτέλεσμα προγράμματος ήταν το ίδιο και στις δύο περιπτώσεις. Υπάρχει όμως μία περίπτωση που οι δύο

μηχανισμοί επιφέρουν διαφορετικά αποτελέσματα. Αν έχουμε μία διαδικασία που αυξάνει κατά ένα την τιμή δύο παραμέτρων (α <- α + 1, β <- β + 1) και την καλέσουμε με την ίδια μεταβλητή (ΚΑΛΕΣΕ Αύξηση(α, α)) τότε στο μηχανισμό με αναφορά το α θα αυξηθεί κατά δύο ενώ στο μηχανισμό με αντιγραφή κατά 1! Αυτό γίνεται επειδή δημιουργούνται δύο αντίγραφα του α, αυξάνονται και τα δύο κατά ένα και τελικά επιστρέφουν την καινούργια τιμή τους στο αρχικό α.

Καλύτερα να αποφεύγεται η κατασκευή ασκήσεων που χρησιμοποιούν αυτήν τη δυσνόητη συμπεριφορά.