Επανάληψη Πληροφορικής Γ' Γυμνασίου

Επίλυση προβλημάτων στον υπολογιστή

Τι έχουμε: Ένα πρόβλημα

Τι θέλουμε να κάνουμε: Να χρησιμοποιήσουμε τον υπολογιστή για να λύσουμε το πρόβλημα.

Ορισμός προβλήματος: Κάθε ζήτημα που τίθεται προς επίλυση, κάθε κατάσταση που μας απασχολεί και πρέπει να αντιμετωπιστεί.

Για παράδειγμα πρόβλημα μπορεί να είναι: ο υπολογισμός της υποτείνουσας ενός ορθογωνίου τριγώνου, η νίκη σε μια παρτίδα σκάκι κ.ά.

Η λύση ενός προβλήματος (αυτό ονομάζεται και **ζητούμενο**), απαιτεί από εμάς δυο βασικές ενέργειες:

1. Να κατανοήσουμε το πρόβλημα
2. Να καταγράψουμε τα δεδομένα του προβλήματος.

Για παράδειγμα για να νικήσουμε στο σκάκι θα πρέπει πρώτα να γνωρίζουμε τους κανόνες του παιχνιδιού (κατανόηση του προβλήματος) και επίσης να ξέρουμε τις τρέχουσες θέσεις που έχουν τα πιόνια τόσο τα δικά μας όσο και του αντιπάλου (καταγραφή των δεδομένων).

Αφού λοιπόν κατανοήσουμε το πρόβλημα και καταγράψουμε τα δεδομένα του, η επόμενη ενέργεια είναι να περιγράψουμε τη λύση του προβλήματος. Η περιγραφή της λύσης θα πρέπει να είναι πολύ συγκεκριμένη και αναλυτική και θα πρέπει να δίνεται σε μια σειρά από βήματα. Αύτη η αναλυτική περιγραφή της λύσης του προβλήματος ονομάζεται *αλγόριθμος*.

Ορισμός αλγορίθμου: Αλγόριθμο ονομάζουμε τη σαφή και ακριβή περιγραφή μιας σειράς ξεχωριστών οδηγιών-βημάτων, με σκοπό την επίλυση ενός προβλήματος.

Για παράδειγμα ας θεωρήσουμε το ακόλουθο πρόβλημα:

«Περιγράψτε σε ένα μικρό παιδί πως θα δημιουργήσει με τις πατούσες του ένα τετράγωνο στην άμμο».

Μήπως όμως λείπει κάτι από την περιγραφή του προβλήματος; Λείπει ο προσδιορισμός του μήκους της πλευράς του τετραγώνου. Υπάρχουν μεγάλα ή μικρότερα τετράγωνα, το καθένα έχει το δικό του μήκος πλευράς. Εμείς ποιο από όλα πρέπει να σχεδιάσουμε; Επομένως η ορθή διατύπωση είναι:

«Περιγράψτε σε ένα μικρό παιδί πως θα δημιουργήσει με τις πατούσες του ένα τετράγωνο στην άμμο που κάθε πλευρά του θα έχει μήκος 5 βήματα».

Έτσι για το πρόβλημα της σχεδίασης του τετραγώνου, το μήκος της πλευράς που πρέπει να έχει είναι *το δεδομένο του προβλήματος*.

Ας περάσουμε τώρα στην διαδικασία επίλυσης του προβλήματος. Ας προσπαθήσουμε να καταγράψουμε τον τρόπο με τον οποίο θα περπατήσει το παιδί για να σχηματιστεί ένα τετράγωνο με κάθε πλευρά να είναι 5 βήματα:

Βήμα 1: Περπάτησε ευθεία μπροστά 5 βήματα.

Βήμα 2. Στρίψε δεξιά κατά 90 μοίρες.

Βήμα 3: Περπάτησε ευθεία μπροστά 5 βήματα.

Βήμα 4. Στρίψε δεξιά κατά 90 μοίρες.

Βήμα 5: Περπάτησε ευθεία μπροστά 5 βήματα.

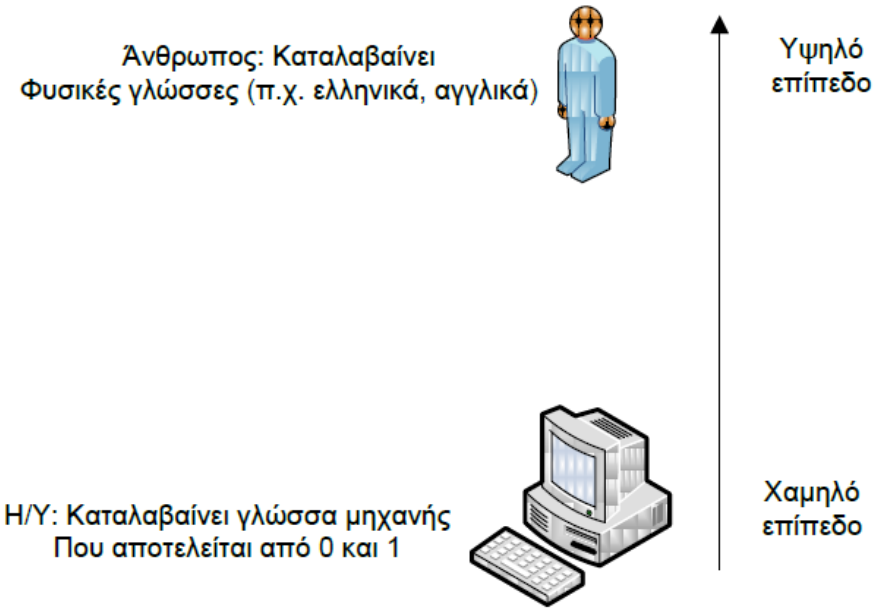
Βήμα 6. Στρίψε δεξιά κατά 90 μοίρες.

Βήμα 7: Περπάτησε ευθεία μπροστά 5 βήματα.

Αυτός λοιπόν ο αναλυτικός τρόπος περιγραφής της λύσης του προβλήματος είναι ο αλγόριθμος για το πρόβλημα. **Ένας αλγόριθμος αποτελείται πάντα από βήματα**

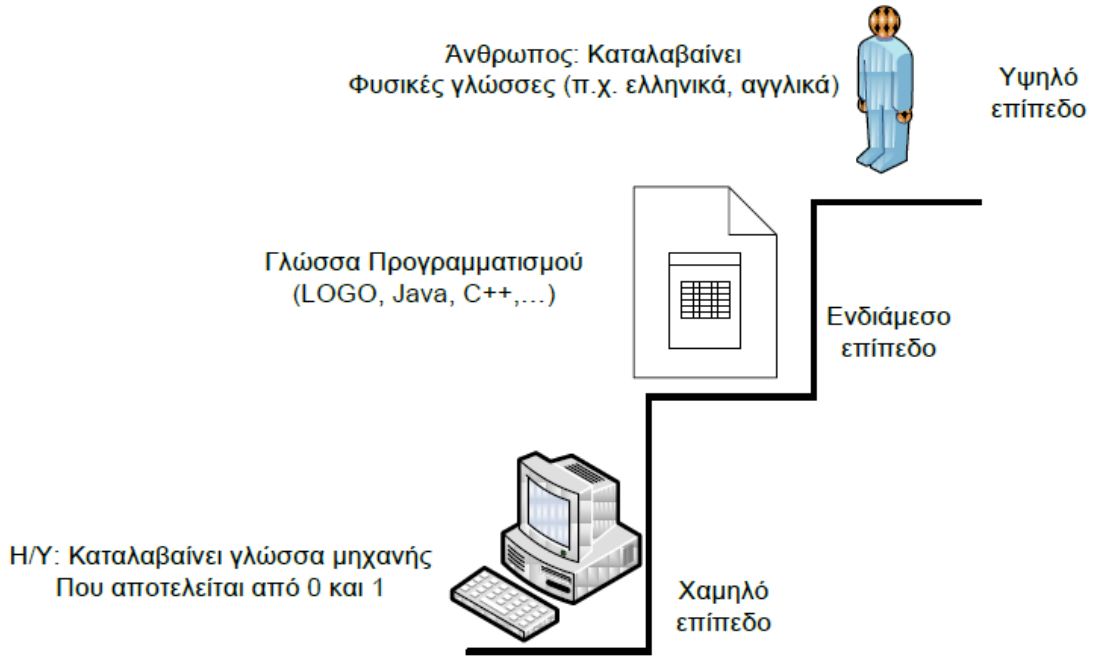
Γλώσσες προγραμματισμού

Αν λέγαμε σε ένα μικρό παιδί τον αλγόριθμο για το τετράγωνο στην άμμο, τότε αυτό θα το καταλάβαινε και θα μπορούσε να ακολουθήσει τις οδηγίες. Σίγουρα όμως όχι και ένας υπολογιστής. Το πρόβλημα είναι ότι ο υπολογιστής καταλαβαίνει μόνο την ψηφιακή γλώσσα που αποτελείται από 0 και 1 και τίποτα περισσότερο. Επομένως για να δώσουμε στον υπολογιστή τις απαιτούμενες οδηγίες θα πρέπει να είναι διατυπωμένες σε αυτή την πολύ χαμηλού επιπέδου γλώσσα, την γλώσσα που αποτελείται μόνο από 0 και 1. Αυτή η γλώσσα ονομάζεται **γλώσσα μηχανής.** Έτσι λοιπόν βρισκόμαστε στην ακόλουθη κατάσταση:



Εικόνα 1: Άνθρωπος και υπολογιστής

Επειδή λοιπόν υπάρχει αυτή η μεγάλη διαφορά στο επίπεδο της γλώσσας που μιλάει ο άνθρωπος και της γλώσσας που καταλαβαίνει ο υπολογιστής έχει εφευρεθεί μια τεχνική που βοηθάει να μειωθεί αυτή η απόσταση, **οι γλώσσες προγραμματισμού**. Μπορούμε να σκεφτούμε τις γλώσσες προγραμματισμού όπως ακριβώς μια σκάλα μας βοηθάει να κατέβουμε σε ένα υπόγειο μοιράζοντας τη διαφορά ύψους σε μικρότερα κομμάτια, τα σκαλοπάτια.

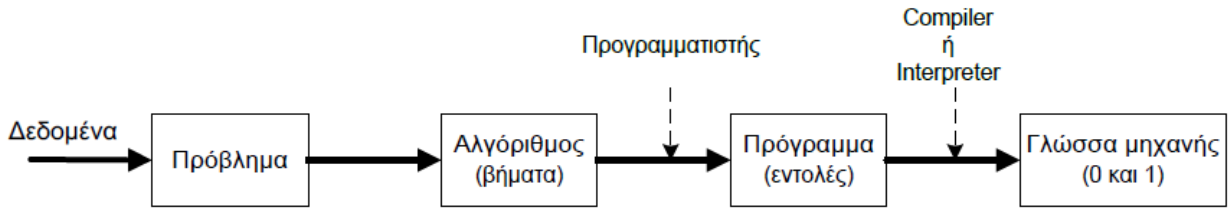


Εικόνα 2: Η γλώσσα προγραμματισμού ως ενδιάμεσο βήμα

Όπως ένας αλγόριθμος αποτελείται από βήματα έτσι και ένα πρόγραμμα αποτελείται από *εντολές*. Επίσης όπως υπάρχουν πολλές φυσικές γλώσσες (ελληνικά, αγγλικά κτλ) έτσι υπάρχουν και πολλές γλώσσες προγραμματισμού (Java, C, C++, Logo, PHP, Pascal, Basic κ.ά). Δείτε στο σχολικό βιβλίο το πρόγραμμα στη γλώσσα C που φαίνεται στην εικόνα 1.2 στη σελίδα 182. Κάθε γλώσσα προγραμματισμού έχει τα ακόλουθα χαρακτηριστικά:

- **Αλφάβητο**. Το σύνολο των χαρακτήρων που χρησιμοποιούνται από τη γλώσσα
- **Λεξιλόγιο**. Το σύνολο των λέξεων που αναγνωρίζει η γλώσσα
- **Συντακτικό**. Το σύνολο των κανόνων που πρέπει να ακολουθούμε για να συνδέουμε τις λέξεις σε προτάσεις.

Επομένως μετά και από αυτό έχουμε το ακόλουθο διάγραμμα που περιγράφει πως μπορούμε να χρησιμοποιήσουμε τον υπολογιστή για να λύσουμε ένα πρόβλημα:



Εικόνα 3: Διαδικασία επίλυσης προβλήματος στον Η/Υ

Το μεγάλο πλεονέκτημα που μας δίνουν οι γλώσσες προγραμματισμού είναι οτι η μετατροπή του προγράμματος σε 0 και 1 γίνεται μέσω αυτόματων εργαλείων που διαθέτει ο Η/Υ. Αυτά είναι:

- Μεταγλωττιστές (compilers).
- Διερμηνείς (interpreters).

Με άλλα λόγια η δική μας εργασία περιλαμβάνει ουσιαστικά 3 βήματα:

1. Κατανόηση του προβλήματος και καταγραφή των δεδομένων του
2. Περιγραφή του αλγορίθμου επίλυσης του προβλήματος
3. Συγγραφή προγράμματος με βάση κάποια γλώσσα προγραμματισμού που υλοποιεί τον αλγόριθμο.

Όταν γράφουμε ένα πρόγραμμα μπορεί να κάνουμε λάθη. Αυτά χωρίζονται σε δύο κατηγορίες:

Συντακτικά λάθη. Αυτά συμβαίνουν όταν αυτό που γράφουμε για πρόγραμμα δεν είναι σύμφωνο με τους κανόνες γραμματικής της γλώσσας, κάτι που δεν βγάζει νόημα ή δεν υπάρχει καν στη συγκεκριμένη γλώσσα προγραμματισμού.

Λογικά λάθη. Σε αυτή την περίπτωση μπορεί αυτό που γράφουμε να είναι ορθό σύμφωνα με τους κανόνες της γλώσσας προγραμματισμού αλλά δεν αποτελεί λύση για το πρόβλημα που θέλουμε να λύσουμε. Για παράδειγμα αν θέλουμε να κάνουμε πρόσθεση δύο αριθμών αλλά αντί για 5 + 2 γράψουμε 5 – 2 τότε αντί για το άθροισμα θα πάρουμε τη διαφορά. Το πρόγραμμα μεν είναι σωστό αλλά δεν κάνει αυτό που θέλουμε.

Εντολές LOGO

Στα πλαίσια του μαθήματος εξετάσαμε τη γλώσσα προγραμματισμού LOGO μέσω του προγραμματιστικού περιβάλλοντος MicroWorlds Pro. Όπως συμβαίνει σε όλες τις γλώσσες έτσι και στη LOGO υπάρχουν *εντολές*. Πρέπει να θυμόμαστε τι κάνει η κάθε εντολή καθώς και *πως συντάσσεται*.

Οι εντολές στη LOGO μπορεί να είναι είτε «ανεξάρτητες» που είναι αυτές που στο MicroWorlds Pro γράφουμε στο γκρι πλαίσιο, είτε ομαδοποιημένες σε *διαδικασίες* που είναι αυτές που γράφουμε στο λευκό πλαίσιο στα δεξιά. Οι ανεξάρτητες εντολές εκτελούνται αμέσως μόλις πατήσουμε το πλήκτρο Enter, ενώ οι εντολές των διαδικασιών εκτελούνται όταν *καλέσουμε* τη διαδικασία. Η κλήση μιας διαδικασίας γίνεται γράφοντας το όνομά της στο γκρι πλαίσιο και πατώντας το πλήκτρο Enter.

Στη συνέχεια παραθέτουμε τις σημαντικότερες από τις εντολές που έχουμε δει στο μάθημα.

Εντολή δείξε

Η εντολή δείξε χρησιμοποιείται στη LOGO για δύο σκοπούς:

Προβολή μηνυμάτων στην οθόνη. Π.χ. η εντολή

```
Δείξε [Καλημέρα κόσμε]
```

Θα εμφανίσει το μήνυμα:

```
Καλημέρα κόσμε
```

Υπολογισμός αριθμητικών πράξεων π.χ. η εντολή

```
Δείξε 5 + 8
```

Θα εμφανίσει το αποτέλεσμα της πρόσθεσης του 5 με το 8 δηλαδή θα δείξει

```
13
```

Τα σύμβολα για τις πράξεις είναι:

Σύμβολο	Πράξη
Πρόσθεση	+
Αφαίρεση	-
Πολλαπλασιασμός	*
Διαίρεση	/

Στην περίπτωση των μηνυμάτων δεν θα πρέπει να ξεχνάμε να περικλείουμε το μήνυμα που θέλουμε να εμφανιστεί μέσα σε αγκύλες [ ] . Στην περίπτωση των αριθμητικών υπολογισμών δεν θα πρέπει να ξεχνάμε την προτεραιότητα των πράξεων. Πρώτα εκτελούνται πολλαπλασιασμοί και διαιρέσεις και μετά προσθέσεις και αφαιρέσεις εκτός αν υπάρχουν παρενθέσεις. Έτσι έχουμε τα ακόλουθα παραδείγματα:

Εντολή	Αποτέλεσμα
`Δειξε 1 + 5 * 3`	16
`Δειξε 1 + 5 * 3 - 6`	10
`Δειξε  (1 + 5) * 3`	18
`Δειξε 1 + 5 * 3 - 6 / 3`	14 (πρώτα θα γίνει το 5*3 μετά το 6/3 και μετά το 1 + 15 – 2)

Εντολές για σχεδίαση σχημάτων στην οθόνη

Η LOGO διαθέτει εντολές για σχεδίαση σχεδίων στην οθόνη με τη χρήση της χελώνας. Η χελώνα μπορεί να κινείται, να στρίβει και να ανεβοκατεβάζει το στυλό γραφής. Οι εντολές με τις οποίες ελέγχουμε τη χελώνα είναι:

Εντολή	Εξήγηση	Παράδειγμα
στκ	κατεβάζει το στυλό	`στκ`
στα	ανεβάζει το στυλό	`στα`
μπ <αριθμός εικονοστοιχείων>	κίνηση εμπρός	`μπ 100`
πι <αριθμός εικονοστοιχείων>	κίνηση πίσω	`πι 100`
δε <μοίρες>	στροφή δεξία	`δε 90`
αρ <μοίρες>	στροφή αριστερά	`αρ 90`
σβγ	σβήσιμο όλων των γραφικών	`σβγ`

Για παράδειγμα με τις ακόλουθες εντολές μπορούμε να σχεδιάσουμε ένα τετράγωνο πλευράς 100

```
στκ
μπ 100
δε 90
μπ 100
δε 90
μπ 100
δε 90
μπ 100
δε 90
```

Δομή επανάληψης

Τη χρησιμοποιούμε όταν θέλουμε να επαναλάβουμε ένα σύνολο εντολών. Η σύνταξη της εντολής είναι

```
Επανάλαβε <πόσες φορές> [ <εντολές που θα επαναληφθούν> ]
```

Για παράδειγμα αν θέλουμε να γράψουμε 10 φορές το όνομα Νίκος στην οθόνη μπορούμε να δώσουμε την εντολή

```
Επανάλαβε 10 [δείξε [Νίκος]]
```

Παρατηρώντας το πρόγραμμα που σχεδιάζει το τετράγωνο βλέπουμε ότι μπορούμε να το χωρίσουμε σε ομάδες εντολών που επαναλαμβάνονται

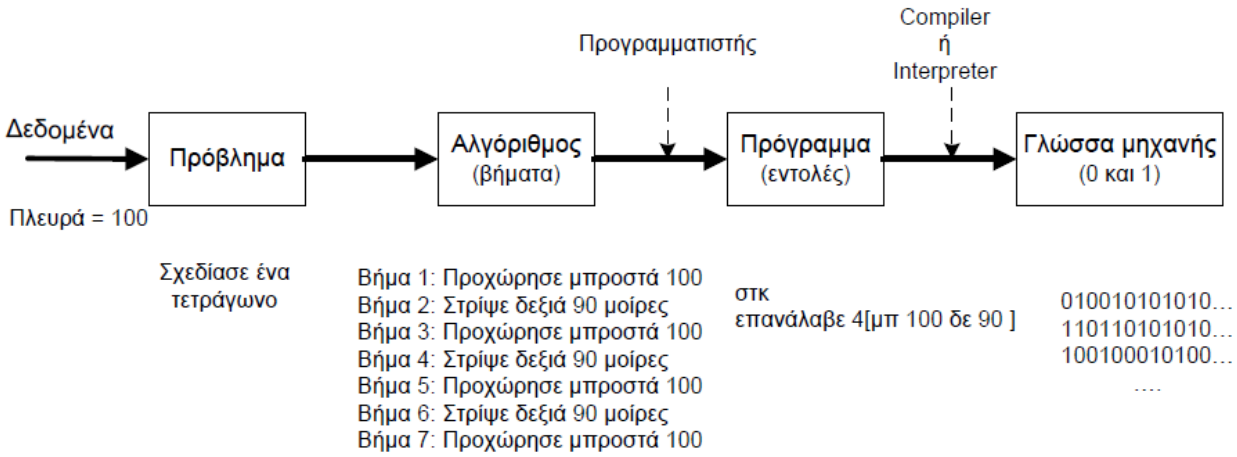
Πρόγραμμα	Εξήγηση
στκ	Κατέβασμα κάτω του στυλό της χελωνας
μπ 100	Πρώτη επανάληψη
δε 90	
μπ 100	Δεύτερη επανάληψη
δε 90	
μπ 100	Τρίτη επανάληψη
δε 90	
μπ 100	Τέταρτη επανάληψη
δε 90	

Έτσι λοιπόν μπορούμε να εξοικονομήσουμε εντολές γράφοντας:

```
στκ
επανάλαβε 4 [ μπ 100 δε 90]
```

Οι εντολές `μπ 100` και `δε 90` θα επαναληφθούν 4 φορές, όσες δηλαδή χρειάζεται για να σχεδιαστεί το τετράγωνο.

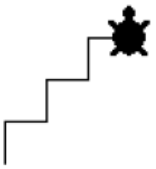
Με αυτό υπόψη μπορούμε να δούμε τώρα πως αντιστοιχεί το παράδειγμα της σχεδίασης του τετραγώνου στο γενικό σχήμα επίλυσης προβλήματος από τον υπολογιστή



Εικόνα 4: Σχεδίαση του τετραγώνου στον υπολογιστή

Κάτω από κάθε κουτάκι φαίνεται ποια ακριβώς ήταν η ενέργεια που έπρεπε να γίνει ώστε σταδιακά να λυθεί το πρόβλημα της σχεδίασης του τετραγώνου.

Ας δούμε τώρα ένα άλλο παράδειγμα. Έστω ότι θέλουμε να σχεδιάσουμε το ακόλουθο σχέδιο



Αν παρατηρήσουμε προσεκτικά το σχέδιο της σκάλας, βλέπουμε ότι αποτελείται από 3 σκαλοπάτια. Επομένως μια ιδέα είναι να γράψουμε τις εντολές που σχεδιάζουν ένα σκαλοπάτι και μετά απλά να χρησιμοποιήσουμε την εντολή επανάλαβε για να σχεδιάσουμε 3 επαναλήψεις τους. Ένα σκαλοπάτι σχεδιάζεται από τις ακόλουθες εντολές:

```
μπ 20
δε 90
μπ 20
αρ 90
```

Επομένως για να σχεδιάσουμε όλη τη «σκάλα» που αποτελείται από 3 «σκαλοπάτια» πρέπει να γράψουμε:

```
επανάλαβε 3 [μπ 20 δε 90 μπ 20 αρ 90]
```

Διαδικασίες

Όπως είπαμε και στην αρχή οι εντολές στο MicroWorlds Pro μπορούν είτε να γραφούν ανεξάρτητα στο γκρι πλαίσιο, είτε ως ομάδες εντολών, τις διαδικασίες. Μια διαδικασία χωρίζεται στα ακόλουθα δύο μέρη:

Ορισμός της διαδικασίας. Είναι οι εντολές που απαρτίζουν τη διαδικασία, αυτό που γράφουμε στο λευκό πλαίσιο στα δεξιά στο MicroWorlds Pro. Γράφοντας τον ορισμό της διαδικασίας ορίζουμε τι θα κάνει η διαδικασία, ωστόσο μέχρι να την καλέσουμε δεν κάνει τίποτα.

Κλήση της διαδικασίας. Έχοντας προηγουμένως ορίσει μια διαδικασία για να την ενεργοποιήσουμε θα πρέπει να την καλέσουμε. Αυτό γίνεται γράφοντας το όνομά της στο γκρι πλαίσιο και πατώντας το πλήκτρο Enter.

Ο ορισμός μιας διαδικασίας έχει την ακόλουθη γενική μορφή σύνταξης:

```
για <όνομα διαδικασίας>
      εντολή1
      εντολή2
      …
      εντολή ν
τέλος
```

Αν υποθέσουμε για παράδειγμα ότι θέλουμε να γράψουμε μια διαδικασία που θα εμφανίζει ένα τετράγωνο πλευράς 100 στην οθόνη τότε θα γράψουμε:

```
για τετράγωνο
      στκ
      επανάλαβε 4 [μπ 100 δε 90]
τέλος
```

Έχοντας γράψει τον ορισμό της διαδικασίας για να εκτελεστούν οι εντολές και να σχεδιαστεί τελικά το τετράγωνο απομένει να την καλέσουμε. Παρατηρούμε ότι το όνομά της διαδικασίας είναι η λέξη *τετράγωνο*, επομένως θα πάμε στο γκρι πλαίσιο και θα γράψουμε

```
τετράγωνο
```

Είναι ενδιαφέρον πως μπορούμε να διαλέξουμε όποιο όνομα θέλουμε για τη διαδικασία. Για παράδειγμα ας υποθέσουμε ότι έχουμε τις ακόλουθες διαδικασίες:

```
για ορθογώνιο
      στκ
      επανάλαβε 4 [μπ 100 δε 90]
τέλος
```

Το όνομα της διαδικασίας είναι *ορθογώνιο*

```
για Νίκος
      στκ
      επανάλαβε 4 [μπ 100 δε 90]
τέλος
```

Το όνομα της διαδικασίας είναι *Νίκος*

Τι ακριβώς κάνουν αυτές οι διαδικασίες; Αν κοιτάξουμε προσεκτικά τις εντολές που περιλαμβάνουν θα διαπιστώσουμε ότι σχεδιάζουν ένα τετράγωνο. Στην πραγματικότητα αυτό που άλλαξε ήταν απλώς το όνομα. Άρα τελικά δεν άλλαξε τίποτα. Πως όμως θα καλέσουμε την καθεμία; Πάντα με το όνομα της καθεμίας. Έτσι την πρώτη θα την καλέσουμε γράφοντας *ορθογώνιο* και τη δεύτερη γράφοντας *Νίκος* στο γκρι πλαίσιο.

Μεταβλητές γενικά

Μια μεταβλητή είναι μια περιοχή μνήμης στον υπολογιστή στην οποία μπορούμε να αποθηκεύσουμε δεδομένα όπως αριθμούς, λέξεις κ.ά. Τα βασικά χαρακτηριστικά μιας μεταβλητής είναι:

- Το όνομά της.
- Το περιεχόμενό της.

Μπορούμε λοιπόν να φανταστούμε μια μεταβλητή ως ένα άδειο δοχείο. Το δοχείο έχει ένα περιεχόμενο και μια ταμπέλα. Η ταμπέλα εξεικονίζει το όνομα της μεταβλητής ενώ το περιεχόμενο την τιμή της. Για παράδειγμα αν έχω μια μεταβλητή με όνομα *Ημέρα* τότε μέσα σε αυτή μπορώ να αποθηκεύσω μια τιμή όπως *Δευτέρα* ή μια τιμή όπως *Τρίτη* ή όπως *Σάββατο*. Ωστόσο κάθε φορά η τιμή που θα έχει η μεταβλητή (το περιεχόμενό της δηλαδή), θα είναι αυτό που έχω βάλει την τελευταία φορά, ότι υπήρχε προηγουμένως χάνεται. Επίσης η τιμή της μεταβλητής δεν έχει καμία σχέση με το όνομά της. Δηλαδή στην μεταβλητή *Ημέρα* θα μπορούσαμε να αποθηκεύσουμε οτιδήποτε όπως το όνομά μου *Νίκος* ή ακόμα και αριθμούς όπως 10, 3.14 κτλ.

Χρονική στιγμή	**Όνομα μεταβλητής**	**Ενέργεια που κάνω**	**Τιμή της μεταβλητής**	**Σχόλια**
0	Ημέρα	-		Στην αρχή η μεταβλητή δεν έχει καμία τιμή
1	Ημέρα	Βάζω *Δευτέρα*	*Δευτέρα*	Τώρα η μεταβλητή έχει την τιμή *Δευτέρα*
2	Ημέρα	Βάζω *Τρίτη*	*Τρίτη*	Τώρα η μεταβλητή έχει την τιμή *Τρίτη* η προηγούμενη τιμή *Δευτέρα* χάθηκε
3	Ημέρα	Βάζω *Νίκος*	*Νίκος*	Τώρα η μεταβλητή έχει την τιμή *Νίκος*, η τιμή *Τρίτη* χάθηκε.
4	Ημέρα	Βάζω *18*	*18*	Η μεταβλητή έχει την τιμή *18*. Δεν με απασχολεί καθόλου ότι το όνομα της μεταβλητής είναι *Ημέρα* μπορώ να αποθηκεύω ότι θέλω.

Ας δούμε τώρα στην πράξη πως μπορούμε να χρησιμοποιήσουμε τις μεταβλητές. Στη LOGO έχουμε δύο τρόπους:

- Ως παραμέτρους στις διαδικασίες.
- Χρησιμοποιώντας την εντολή *κάνε*.

Μεταβλητές ως παράμετροι των διαδικασιών

Θυμόμαστε πως ένα βασικό χαρακτηριστικό κάθε προβλήματος είναι τα δεδομένα που το περιγράφουν και στην περίπτωση της σχεδίασης ενός τετραγώνου δεδομένο είναι το μήκος της πλευράς του. Εμείς έχουμε δει πως μπορούμε να φτιάξουμε μια διαδικασία για την σχεδίαση ενός τετραγώνου πλευράς 100. Θα μπορούσαμε όμως άραγε να σχεδιάσουμε ένα τετράγωνο με μήκος πλευράς όση θέλουμε εμείς κάθε φορά; Αυτό γίνεται με τον ακόλουθο τρόπο:

```
για τετράγωνο :πλευρά
      στκ
      επανάλαβε 4 [μπ :πλευρά δε 90]
τέλος
```

Μεταβλητή πλευρά

Βλέπουμε ότι μετά το όνομα της διαδικασίας ακολουθεί το όνομα μιας μεταβλητής με την άνω-κάτω τελεία μπροστά από το όνομά της. Αν προσέξουμε τις εντολές της διαδικασίας βλέπουμε ότι αντί για *μπ 100* μέσα στο επανάλαβε, γράφουμε *μπ :πλευρά*. Αυτό σημαίνει ότι η χελώνα αντί να κινηθεί μπροστά κατά 100, θα κινηθεί όσο είναι η τιμή της μεταβλητής *πλευρά*. Αν η τιμή της μεταβλητής *πλευρά* είναι 150 θα κινηθεί 150, αν είναι 20 θα κινηθεί 20 κ.ό.κ. Επομένως η διαδικασία μπορεί τώρα να σχεδιάζει τετράγωνα όσης πλευράς θέλουμε. Κάθε φορά το τετράγωνο θα έχει τόση πλευρά όση είναι η τιμή που θα έχουμε δώσει στην μεταβλητή *πλευρά*.

Ας δούμε τώρα πως μπορούμε να χρησιμοποιήσουμε τη διαδικασία δίνοντας ταυτόχρονα τιμή στην μεταβλητή *πλευρά*. Αυτό γίνεται αν καλέσουμε τη διαδικασία με τον ακόλουθο τρόπο (γράφοντας στο γκρι πλαίσιο):

```
τετράγωνο 200
```

…δίνουμε τιμή 200 στην μεταβλητή *πλευρά*

Καλούμε τη διαδικασία τετράγωνο και…

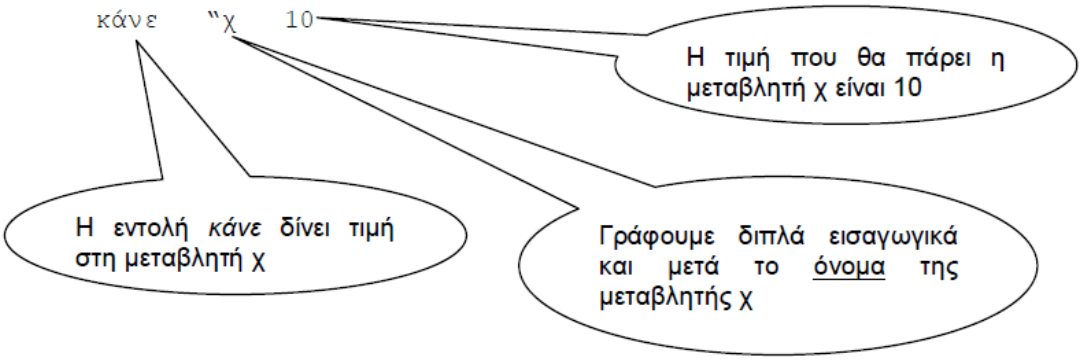
Αυτό που κάνει η παραπάνω εντολή είναι να καλεί την διαδικασία *τετράγωνο* και ταυτόχρονα να δίνει τιμή στην μεταβλητή *πλευρά* το 200. Επομένως το αποτέλεσμα αυτής της εντολής θα είναι η χελώνα να σχεδιάσει ένα τετράγωνο με πλευρά 200. Στον επόμενο πίνακα βλέπουμε διάφορα παραδείγματα κλήσης της διαδικασίας πλευρά:

Χρονική στιγμή	**Εντολή που πληκτρολογούμε στο γκρι πλαίσιο**	**Αποτέλεσμα της εντολής**
1	τετράγωνο 200	σχεδίαση τετραγώνου με πλευρά 200
2	τετράγωνο 50	σχεδίαση τετραγώνου με πλευρά 50
3	τετράγωνο 150	σχεδίαση τετραγώνου με πλευρά 150

Βλέπουμε δηλαδή ότι το αποτέλεσμα δεν θα είναι κάθε φορά το ίδιο, αλλά εξαρτάται από την τιμή που θα δώσουμε στην μεταβλητή πλευρά όταν θα καλέσουμε τη διαδικασία.

Εντολή κάνε

Ο δεύτερος τρόπος με τον οποίο μπορούμε να χρησιμοποιήσουμε τις μεταβλητές στη LOGO είναι μέσω της εντολής *κάνε*. Η εντολή *κάνε* αναλαμβάνει να δώσει τιμή σε μια μεταβλητή. Για παράδειγμα έχουμε την ακόλουθη εντολή:



Ας δούμε στον ακόλουθο πίνακα τι ακριβώς θα συμβεί γράφοντας μερικές εντολές *κάνε*

Χρονική στιγμή	**Εντολή**	**Αποτέλεσμα**
1	`κάνε "χ 10`	Η μεταβλητή χ παίρνει την τιμή 10
2	`κάνε "χ 20`	Η μεταβλητή χ παίρνει την τιμή 20 (το 10 χάνεται)
3	`κάνε "χ "Νίκος`	Η μεταβλητή χ παίρνει την τιμή Νίκος (το 20 χάνεται)
4	`κάνε "Ημέρα "Δευτέρα`	Η μεταβλητή *Ημέρα* παίρνει την τιμή *Δευτέρα*
5	`κάνε "Ημέρα 50`	Η μεταβλητή *Ημέρα* παίρνει την τιμή *50*

Από τον παραπάνω πίνακα προκύπτουν μερικά ενδιαφέροντα συμπεράσματα:

- Όπως έχουμε δει κάθε φορά η μεταβλητή διατηρεί την τελευταία της τιμή.
- Σε μια μεταβλητή μπορούμε να δώσουμε όχι μόνο αριθμητικές τιμές αλλά και τιμές με χαρακτήρες (γράμματα και σύμβολα). **Σε αυτή την περίπτωση όμως πρέπει να βάλουμε μπροστά διπλά εισαγωγικά** (δείτε τις χρονικές στιγμές 3 και 4). Δηλαδή πρέπει να γράψουμε *"Νίκος* ή *"Δευτέρα* (με διπλά εισαγωγικά στην αρχή), επειδή πρόκειται για λέξεις. Αν όμως είναι αριθμοί (όπως στην εντολή `κάνε "χ 10`), τότε δεν χρειάζονται διπλά εισαγωγικά.
- Μπορούμε να βάλουμε ότι θέλουμε σαν περιεχόμενο μιας μεταβλητής ακόμα και αν αυτό δεν σχετίζεται με το νόημα του ονόματός της. Για παράδειγμα στη χρονική στιγμή 5 δίνουμε στην μεταβλητή *Ημέρα* την τιμή 50.

Δομή ελέγχου

Μέχρι στιγμής έχουμε δει πως μπορούμε να γράφουμε εντολές LOGO για να κάνουμε διάφορα πράγματα. Ωστόσο σε κάθε περίπτωση οι εντολές είχαν το χαρακτηριστικό ότι εκτελούνταν η μία

κατόπιν της άλλης με τη σειρά που ήταν γραμμένες. Κάτι τέτοιο όμως δεν είναι πάντοτε αυτό που θέλουμε να κάνουμε.

Για παράδειγμα ας υποθέσουμε ότι θέλουμε να υπολογίσουμε την απόλυτη τιμή ενός αριθμού. Η απόλυτη τιμή ενός αριθμού είναι η τιμή του αριθμού με θετικό πρόσημο πάντα. Έτσι αν για παράδειγμα έχουμε τον αριθμό -5 η απόλυτη τιμή του είναι 5, για το -3 είναι 3 και για το 10 είναι 10.

Έχουμε λοιπόν το ακόλουθο πρόβλημα:

«Αν σου δώσω έναν αριθμό, βρες μου την απόλυτη τιμή του»

Ας δοκιμάσουμε να φτιάξουμε έναν αλγόριθμο που να λύνει το πρόβλημα.

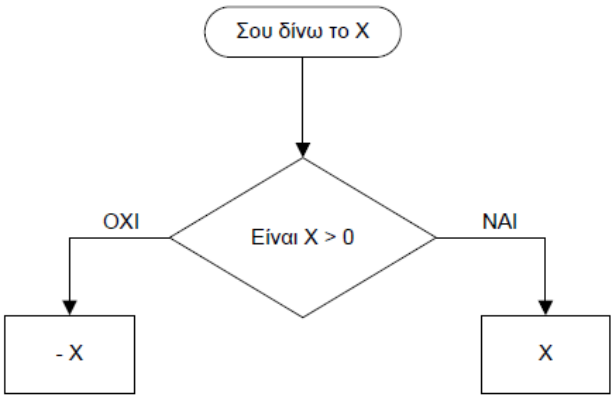
Βήμα 1: Έλεγξε αν ο αριθμός έχει θετικό(+) ή αρνητικό(-) πρόσημο.

Βήμα 2: **Αν** ο αριθμός έχει θετικό πρόσημο τότε η απόλυτη τιμή του είναι ο ίδιος ο αριθμός

Βήμα 3: **Διαφορετικά** η απόλυτη τιμή του είναι ο αριθμός αλλά με θετικό πρόσημο (+) αντί για αρνητικό (-).

Παρατηρούμε δηλαδή ότι το βήμα 1 καθορίζει αν θα εκτελεστεί το βήμα 2 ή το βήμα 3. Αν το πρόσημο του αριθμού είναι θετικό τότε θα εκτελεστεί το βήμα 2 και όχι το βήμα 3. Στην αντίθετη περίπτωση θα εκτελεστεί το βήμα 3.

Διαγραμματικά αυτό θα μπορούσαμε να το αναπαραστήσουμε με το ακόλουθο σχήμα:



Αρχικά λαβαίνουμε έναν αριθμό Χ, κατόπιν ελέγχουμε αν είναι θετικός ή αρνητικός συγκρίνοντάς τον με το μηδέν. Αν είναι μεγαλύτερος του μηδενός τότε ο έλεγχος Χ>0 είναι *ΝΑΙ,* ακολουθούμε το δεξιό βέλος και η απόλυτη τιμή είναι ο ίδιος ο αριθμός. Αν όμως δεν είναι μεγαλύτερος του μηδενός τότε ο έλεγχος Χ>0 είναι *ΟΧΙ* άρα ακολουθούμε το αριστερό βέλος και η απόλυτη τιμή είναι το –Χ (γράφουμε –Χ διότι εφόσον ο αριθμός είναι αρνητικός άρα με το μείον μπροστά γίνεται θετικός πχ. – (-3)=3) .

Ας δούμε ένα παράδειγμα. Έστω λοιπόν ότι μας δίνουν για Χ το -10. Πρώτα θα συγκρίνουμε το Χ αν είναι μεγαλύτερο από το μηδέν. Αφού για Χ είναι -10, άρα συγκρίνουμε αν -10 > 0 που είναι λάθος. Άρα θα ακολουθήσουμε το αριστερό βέλος για ΟΧΙ. Έτσι θα δώσουμε για απόλυτη τιμή το – (-10) = 10.

Επομένως παρατηρούμε ότι σε μια τέτοια κατάσταση που <u>πρέπει να αποφασίσουμε</u> τι θα κάνουμε υπάρχουν 3 βασικά στάδια:

1. Βάζουμε έναν **έλεγχο** ο οποίος μπορεί να απαντηθεί ή με *ΝΑΙ* ή με *ΟΧΙ.*
2. Δίνουμε οδηγίες τι θα συμβεί αν ο έλεγχος απαντηθεί με ΝΑΙ
3. Δίνουμε οδηγίες τι θα συμβεί αν ο έλεγχος απαντηθεί με ΟΧΙ

Στη συνέχεια μετατρέπουμε τον αλγόριθμο σε πρόγραμμα. Η ακόλουθη διαδικασία υπολογίζει την απόλυτη τιμή ενός αριθμού:

```
για απόλυτη :χ
      ΑνΔιαφορετικά :χ > 0
      [ δειξε :χ ]
      [ δειξε -1 * :χ ]
τέλος
```

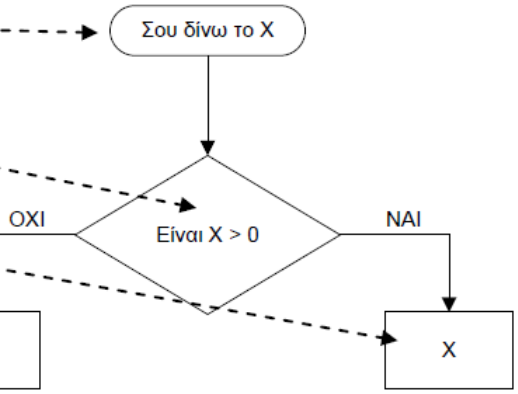
Έλεγχος αν χ >0

Εντολές που θα εκτελεστούν για ΝΑΙ

Εντολές που θα εκτελεστούν για ΟΧΙ

Η εντολή `ΑνΔιαφορετικά` κάνει τον έλεγχο αν χ>0 και αν ο έλεγχος απαντηθεί με *ΝΑΙ* τότε εκτελούνται οι εντολές που βρίσκονται μέσα στο πρώτο ζευγάρι αγκυλών [ ]. Αν ο έλεγχος απαντηθεί με *ΟΧΙ* τότε θα εκτελεστούν οι εντολές που βρίσκονται μέσα στο δεύτερο ζευγάρι αγκυλών [ ]. Έτσι αν ο έλεγχος βγάλει *ΝΑΙ* τότε το πρόγραμμα θα δείξει την τιμή του χ όπως είναι, διαφορετικά θα δείξει την αντίθετη τιμή του χ.

Στη συνέχεια βλέπουμε πως αντιστοιχίζεται κάθε εντολή του προγράμματος με το διάγραμμα ελέγχου που είχαμε δει προηγουμένως.

Πρόγραμμα LOGO	**Διάγραμμα ελέγχου**
για απόλυτη :χ ΑνΔιαφορετικά :χ > 0 [δειξε :χ] [δειξε -1 * :χ] τέλος	

Θα πρέπει να σημειώσουμε πως ο έλεγχος που κάνει η *ΑνΔιαφορετικά* μπορεί να είναι ένα από τα ακόλουθα:

Έλεγχος	**Εξήγηση**
>	Μεγαλύτερο
<	Μικρότερο
=	Ίσο
ανήκει?	Αν η τιμή που εξετάζουμε ανήκει μέσα σε ένα σύνολο.

Η τελευταία περίπτωση με το ανήκει? εξετάζει αν αυτό που ελέγχουμε βρίσκεται μέσα σε ένα σύνολο. Για παράδειγμα έστω η εντολή:

```
Αν Διαφορετικά ανήκει? :χ [Κόκκινο Πράσινο]
```

Αν το χ είναι Κόκκινο τότε ο έλεγχος θα βγάλει *ΝΑΙ*, το ίδιο θα γίνει και αν το χ είναι Πράσινο. Αν όμως το χ είναι Μπλε τότε ο έλεγχος θα βγάλει *ΟΧΙ*, όπως και για οποιοδήποτε άλλο χρώμα εκτός του Κόκκινο και Πράσινο.