Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον

Γ’ Λυκείου

ΚΕΦΑΛΑΙΟ 2

Βασικές έννοιες αλγορίθμων

Χρήστος Μουρατίδης - Έκδοση 2020
mouratx@yahoo.com
http://users.sch.gr/mouratx

Περιεχόμενα

Ορισμός τί είναι αλγόριθμος

Αλγόριθμος : Είναι ένα σύνολο βημάτων, αυστηρά καθορισμένων, κι εκτελέσιμων σε πεπερασμένο χρόνο, που οδηγούν στην επίλυση ενός προβλήματος.

Χαρακτηριστικά ενός σωστού αλγορίθμου (κριτήρια σωστού αλγορίθμου)

Είσοδος	Καμία, μία ή περισσότερες τιμές δεδομένων δίνονται ως είσοδοι.
Έξοδος	Πρέπει να δημιουργεί μία τιμή δεδομένων ως αποτέλεσμα.
Περατότητα	Να τελειώνει, φτάνοντας στο επιθυμητό αποτέλεσμα, σε πεπερασμένο αριθμό βημάτων ή χρόνο. Αν δεν συμβαίνει αυτό τότε είναι απλώς *υπολογιστική διαδικασία*.
Σαφήνεια	Τα επιμέρους βήματα να είναι απλά και σαφή.
Εκτελεσιμότητα	Τα βήματα να είναι έτσι διατυπωμένα έτσι ώστε να μπορούν να εκτελεστούν από τον Η/Υ.
Πληρότητα	Ο αλγόριθμος να προβλέπει κάθε πιθανό ενδεχόμενο κατά την εκτέλεση του (π.χ. μία διαίρεση με το μηδέν).
Αποτελεσματικότητα	Μετά το πέρας της εκτέλεσης των βημάτων να έχει επιτευχθεί ο στόχος (το ζητούμενο/επιθυμητό αποτέλεσμα).

Σπουδαιότητα των αλγορίθμων

Η Πληροφορική μελετά τους αλγόριθμους από διαφορετικές οπτικές γωνίες, οι οποίες είναι οι εξής:

- **Από την πλευρά του Υλικού (hardware):** Η τεχνολογία των επιμέρους μονάδων του υπολογιστή επηρεάζει σημαντικά την **ταχύτητα εκτέλεσης των αλγορίθμων**. Για παράδειγμα, ο αριθμός των πυρήνων του επεξεργαστή, η ταχύτητά του, το μέγεθος της μνήμης RAM καθώς και της κρυφής μνήμης (cache memory), το είδος του δίσκου (σκληρός, SSD κλπ) και άλλα.

- **Από την πλευρά των Γλωσσών Προγραμματισμού:** Η επιλογή της γλώσσας προγραμματισμού που θα χρησιμοποιηθεί για την υλοποίηση του αλγορίθμου, επηρεάζει τη δομή και τον αριθμό των εντολών του αλγορίθμου. Οι λεγόμενες γλώσσες χαμηλού επιπέδου (assembly) είναι ταχύτερες από μία γλώσσα υψηλού επιπέδου (π.χ. Pascal, Basic). Επιπλέον, διαφορές παρατηρούνται και μεταξύ των γλωσσών υψηλού επιπέδου καθώς τα τεχνικά χαρακτηριστικά των νεότερων προσφέρουν περισσότερες δυνατότητες.

- **Από την Θεωρητική πλευρά:** Έχει να κάνει με θέματα **αποδοτικότητας των αλγορίθμων**, όπως για παράδειγμα να βρούμε τον αποδοτικότερο αλγόριθμο που θα επιλύει ένα συγκεκριμένο πρόβλημα.

- **Από την Αναλυτική πλευρά:** Μελετώνται οι **υπολογιστικοί πόροι που απαιτούνται** για την εκτέλεση του αλγορίθμου(φόρτος της CPU και της κύριας μνήμης, χρόνος που απαιτείται για λειτουργίες εισόδου/εξόδου κ.ά).

Τρόποι αναπαράστασης ενός αλγορίθμου

1) **Με ελεύθερο κείμενο :** Αποτελεί έναν *αδόμητο τρόπο περιγραφής*, που μπορεί να καταστρατηγήσει εύκολα τα κριτήρια του σωστού αλγορίθμου (ασάφειες, μη εκτελεσιμότητα κ.λπ.).

2) **Με διαγραμματικές τεχνικές :** Αποτελεί έναν γραφικό τρόπο παρουσίασης των εντολών. Η πιο γνωστή τεχνική είναι το **διάγραμμα ροής** (flow chart) που παλαιότερα ήταν αρκετά δημοφιλής.

3) **Με φυσική γλώσσα κατά βήματα:** Χρειάζεται προσοχή διότι μπορεί να παραβιαστεί το κριτήριο της πληρότητας και σαφήνειας. Δεν είναι προτιμητέος τρόπος.

4) **Με κωδικοποίηση:** Στην περίπτωση αυτή, δημιουργούμε τις εντολές κατευθείαν σε μία *γλώσσα προγραμματισμού* ή σε μία *ψευδογλώσσα* (που μοιάζει με μία πραγματική γλώσσα προγραμματισμού).

Στα παρακάτω παραδείγματα χρησιμοποιούμε μία κωδικοποίηση σε μία μη υπαρκτή δομημένη γλώσσα προγραμματισμού, καλούμενη ως **ψευδογλώσσα**. Φτιάχνοντας τον αλγόριθμο με ψευδογλώσσα, μπορούμε, κατόπιν, να τον μετατρέψουμε χωρίς πολλές μετατροπές σε μία υπαρκτή γλώσσα.

Βασικά στοιχεία αλγορίθμου.

Σταθερές

Αφορούν ποσότητες που δεν μεταβάλλονται κατά τη διάρκεια εκτέλεσης του αλγόριθμου.

Υπάρχουν 3 ειδών :

- ***Αριθμητικές*** π.χ. 123, -8, 3.14
- ***Αλφαριθμητικές*** π.χ. ‘’Γιάννης’’, ‘’Μακεδονίας 12’’ . Λέγονται και λεκτικά ή συμβολοσειρές. Περικλείνονται σε ‘’ (*διπλά ή μονά εισαγωγικά*).
- ***Λογικές*** π.χ. Αληθές/Ψευδές (αγγλικά: True/False).

Μεταβλητές

Αφορούν ποσότητες που μεταβάλλονται κατά τη διάρκεια εκτέλεσης του αλγορίθμου.

Χονδρικά, παριστάνουν μία θέση μνήμης που περιέχει μία τιμή. Το περιεχόμενο αυτής της θέσης (η τιμή της) αλλάζει.

Στις μεταβλητές δίνουμε ένα **όνομα** (αναγνωριστικό) και αυτό χρησιμοποιούμε στον αλγόριθμο. **Η τιμή της αλλάζει με μία εντολή εκχώρησης τιμής** π.χ. `X ← 123` ή `Name ← "Γιάννης"`

Όνομα μεταβλητής ⟶	X	Name
Τιμή της μεταβλητής ⟶	123	"Γιάννης"

Το όνομα της μεταβλητής, από τη στιγμή που θα δηλωθεί στον αλγόριθμο δεν αλλάζει. Το περιεχόμενο (*η τιμή της*), όμως, αλλάζει κατά τη διάρκεια της εκτέλεσης των εντολών του αλγορίθμου.

Κι εδώ, ανάλογα με το είδος δεδομένου που εκχωρείται, η **μεταβλητή διακρίνεται σε**:

- ***Αριθμητική*** π.χ `X ← 123`
- ***Αλφαριθμητική*** π.χ. `Name ← "Γιάννης"`
- ***Λογική*** π.χ. `Έγγαμος ← Αληθές`.

Επιπλέον, οι αριθμητικές διακρίνονται σε :

- *Ακέραιες* , αν δέχονται ακέραια τιμή π.χ. 5 , -12
- *Πραγματικές*, αν δέχονται πραγματική τιμή π.χ. 3.14 , -4.56

Τελεστές

Είναι τα γνωστά σύμβολα που χρησιμοποιούμε στις διάφορες πράξεις.

Είναι 3 ειδών:

- ***Αριθμητικοί τελεστές*** **:** + , - , * (πολλ/σμός), / (διαίρεση) , ^ (ύψωση σε δύναμη), DIV (ακέραια διαίρεση), MOD (ακέραιο υπόλοιπο).

  Π.χ. `X ← 5 * 2`,
  `X ← 6 ^ 2` (=36),
  `X ← 5 DIV 2` (=2 πηλίκο),
  `X ← 5 MOD 2` (=1 υπόλοιπο)

- ***Λογικοί*** : ΚΑΙ, Ή , ΌΧΙ (αγγλικά: AND, OR, NOT).

  Π.χ. `Έγκυρος ← (X >= 1) ΚΑΙ (X <= 20)`

- ***Συγκριτικοί*** : > , >= , < , <= , <> (διάφορο)

Εκφράσεις

Συνδυάζουν *τελεστέους* (σταθερές και μεταβλητές) και *τελεστές*. Ο σκοπός των εκφράσεων είναι να εκτελεστεί μία πράξη και το αποτέλεσμά της να αποδοθεί ως τιμή σε μία μεταβλητή.

Για την **αποτίμηση της τελικής τιμής μίας έκφρασης** παίζει ρόλο η *προτεραιότητα (ιεραρχία) των πράξεων* καθώς και η *ύπαρξη παρενθέσεων*. Ως γνωστόν, η ύπαρξη των παρενθέσεων μεταβάλλει την προτεραιότητα των πράξεων (προηγούνται).

Μία **έκφραση** μπορεί να είναι πολύ **απλή** (να αποτελείται από μία μόνο σταθερά ή μεταβλητή) μέχρι **πολύπλοκη** (π.χ. αριθμητική παράσταση).

Παραδείγματα:

`X ← 1` (απλή, μία σταθερά ως έκφραση)

`X ← ((Y + 5) / 8 + (Y + 10)) MOD 2` (πολύπλοκη αριθμητική παράσταση)

`Έγκυρος ← (Χ >= 1) ΚΑΙ (Χ <= 20)` (πολύπλοκη λογική παράσταση)

Οσοδήποτε πολύπλοκη και να είναι μία έκφραση, δεν πρέπει να ξεχνάμε ότι η **τελική αποτίμησή της είναι μία μόνο τιμή**.

Βασικές εντολές

Στους αλγόριθμους με ψευδογλώσσα, χρησιμοποιούμε τις εξής **εντολές**:

Εντολή εισόδου	**Διάβασε** (εισάγει από μία μονάδα εισόδου ένα δεδομένο)	**Παράδειγμα**: `Διάβασε Χ` Στη μεταβλητή Χ εισάγεται μία τιμή.

Εντολές εξόδου	**Εκτύπωσε**
	(τυπώνει στον εκτυπωτή ένα αποτέλεσμα/τιμή)
	Εμφάνισε
	(εμφανίζει στην οθόνη ένα αποτέλεσμα/τιμή)
		Παράδειγμα 1:
		`Εκτύπωσε Χ`
		Η τιμή της μεταβλητής Χ τυπώνεται στον εκτυπωτή.
		Παράδειγμα 2:
		`Εμφάνισε Χ`
		Η τιμή της μεταβλητής Χ εμφανίζεται στην οθόνη.

Εντολή εκχώρησης τιμής	Μορφή : `Μεταβλητή ← Έκφραση` (Αριστερά βάζουμε το όνομα της μεταβλητής και δεξιά μία έκφραση).	**Παράδειγμα**: `X ← (5 * 2 ) / 100` Η έκφραση δεξιά δίνει ως αποτέλεσμα (αποτίμηση) την τιμή `0,10` και αυτή εκχωρείται(τοποθετείται) στην μεταβλητή Χ.

(Σημ. Ο αγγλικός όρος για την εκχώρηση τιμής είναι *assign value*).

- Με δύο τρόπους λοιπόν, τοποθετούμε τιμές σε μεταβλητές:

  1) Με **εντολή εισόδου** `Διάβασε.`

  2) Με **εντολή εκχώρησης** `Μεταβλητή ← Έκφραση.`

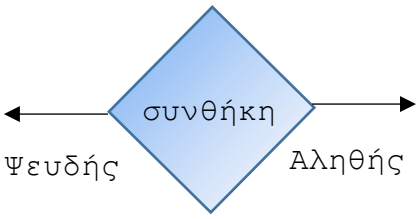
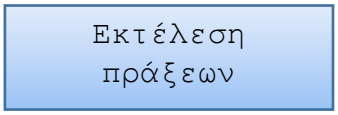
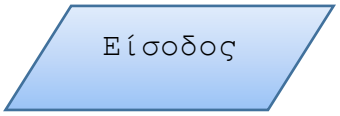
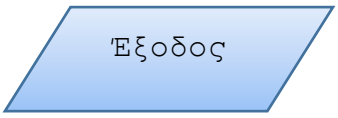
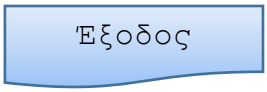
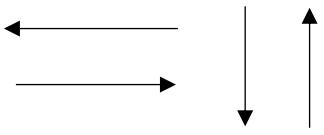
Διάγραμμα Ροής

Το διάγραμμα ροής αναπαριστά με γραφικό τρόπο (*οπτικοποιεί*) τον αλγόριθμο.

Ένα διάγραμμα ροής περιλαμβάνει ένα σύνολο από **γεωμετρικά σχήματα** (π.χ. ορθογώνια, ρόμβους), όπου το καθένα δηλώνει μία συγκεκριμένη **ενέργεια ή λειτουργία**. Τα σχήματα αυτά, **ενώνονται με βέλη**, των οποίων η **φορά δηλώνει και τη κατεύθυνση (ροή) των ενεργειών**.

Παραθέτουμε τα κυριότερα σχήματα:

Σχήμα	Περιγραφή	Παράδειγμα
Έλλειψη	Δηλώνει την **αρχή** και το **τέλος** του αλγορίθμου.	Αρχή Τέλος

Ρόμβος	Δηλώνει μία **συνθήκη ελέγχου** (ερώτηση) με δύο ή περισσότερες εξόδους ως απάντηση.	
Ορθογώνιο	Δηλώνει την **εκτέλεση μίας ή περισσοτέρων πράξεων**.	
Πλάγιο παραλληλόγραμμο	Δηλώνει την **είσοδο δεδομένων ή έξοδο αποτελέσματος.** Μερικές φορές το σχήμα αυτό είναι διαφορετικό ανάλογα με τη συσκευή εισόδου ή εξόδου (*π.χ. έξοδο σε εκτυπωτή ή δίσκο*).	  Ένα πιο εξειδικευμένο σχήμα που υποδηλώνει **έξοδο** αποτελέσματος **σε χαρτί** (*εκτυπωτή*):  Ή **σε δίσκο**: 
Βέλος	Δηλώνει την **ροή εκτέλεσης των εντολών** του αλγορίθμου.	

Παρακάτω, θα δούμε παραδείγματα αναπαράστασης του αλγορίθμου σε ψευδογλώσσα και διάγραμμα ροής.

Κάθε αλγόριθμος που αναπαρίσταται σε ψευδογλώσσα μπορεί ισοδύναμα να μετατραπεί σε διάγραμμα ροής και αντίστροφα.

Αλγοριθμικές δομές

Με τον όρο «Αλγοριθμική δομή» εννοούμε τον τρόπο που θέτουμε τα βήματα (εντολές) στον αλγόριθμο ώστε να έχουμε σωστή και αυστηρή σύνταξη (δομημένη) και να πληρούνται τα χαρακτηριστικά του.

Είναι οι εξής:

Ακολουθιακή δομή	**Οι εντολές εκτελούνται η μία μετά την άλλη**.
Απλής Επιλογής	**Μία ομάδα εντολών εκτελούνται ανάλογα με την τιμή μίας συνθήκης (*απόφαση*).** **Η συνθήκη είναι μία λογική έκφραση με αποτίμηση (*απόφαση*) Αληθές ή Ψευδές**. Αν η αποτίμηση είναι `Αληθής` τότε εκτελείται μία ομάδα εντολών ενώ μπορούμε να καθορίσουμε και μία άλλη ομάδα εντολών να εκτελεστεί αν η αποτίμηση είναι `Ψευδής`. Για παράδειγμα, η συνθήκη «`Κάνει κρύο σήμερα`» είναι μία λογική έκφραση. Αν ισχύει (`Αληθής`) τότε να εκτελεστεί η εντολή `Πάρε το βαρύ παλτό`. Αν δεν ισχύει μπορούμε να καθορίσουμε να εκτελεστεί η εντολή `Πάρε το ελαφρύ μπουφάν`.

Πολλαπλής επιλογής	**Μία ομάδα εντολών εκτελούνται ανάλογα με την τιμή μίας έκφρασης**. Για παράδειγμα, αν ο βαθμός του μαθητή είναι >0 ΚΑΙ <10 τότε να τυπώσει το μήνυμα "Απορρίπτεται", αν είναι μεταξύ >=10 ΚΑΙ <12,5 τότε να τυπώσει το μήνυμα "Μέτρια", αν είναι μεταξύ >=12,5 και <15,5 τότε να τυπώσει το μήνυμα "Καλά" κ.ο.κ. Με άλλα λόγια, ελέγχουμε τον βαθμό του μαθητή σε ποιό εύρος τιμών βρίσκεται κι ανάλογα εκτελείται η αντίστοιχη εντολή εκτύπωσης του μηνύματος.
Εμφωλευμένες διαδικασίες	**Πρόκειται για ειδικές περιπτώσεις πολλαπλών επιλογών, όπου μία δομή επιλογής βρίσκεται μέσα σε μία άλλη δομή επιλογής (*εμφωλευμένη δομή*).** Για παράδειγμα, ελέγχουμε το βάρος ενός μαθητή. Αν είναι >80 κιλά τότε ελέγχουμε αν είναι το ύψος του >1,70 μ. Πρέπει να σημειώσουμε ότι ***εμφωλευμένες δομές μπορεί να έχουμε και στις επαναλήψεις***. Το βάθος της εμφώλευσης μπορεί να είναι μεγάλο ανάλογα με το πρόβλημα.
Επαναληπτική δομή	**Μία ομάδα εντολών εκτελούνται συνεχώς επαναληπτικά ανάλογα με την τιμή μίας συνθήκης (*βρόχος – Loop*).** Για παράδειγμα, η εκτύπωση των ελέγχων επίδοσης όλων των μαθητών. Αν οι μαθητές είναι 180 τότε η διαδικασία εκτύπωσης του ελέγχου θα επαναληφθεί 180 φορές.

Ακολουθιακή δομή

Πολύ *απλά* προβλήματα μπορούν να επιλυθούν με την παράθεση κι εκτέλεση των εντολών *σειριακά*, δηλαδή τη μία μετά την άλλη.

Παράδειγμα: Να γραφεί αλγόριθμος που διαβάζει τη βάση και το ύψος ενός τριγώνου και υπολογίζει και τυπώνει το εμβαδόν του.

Εδώ θα εισαχθούν δύο αριθμητικά δεδομένα (*βάση* και *ύψος*) και θα εξαχθεί σαν αποτέλεσμα (ζητούμενο) το *εμβαδόν*. Για την αποθήκευση των τιμών θέλουμε αντίστοιχα τρεις μεταβλητές.

Με ψευδογλώσσα:

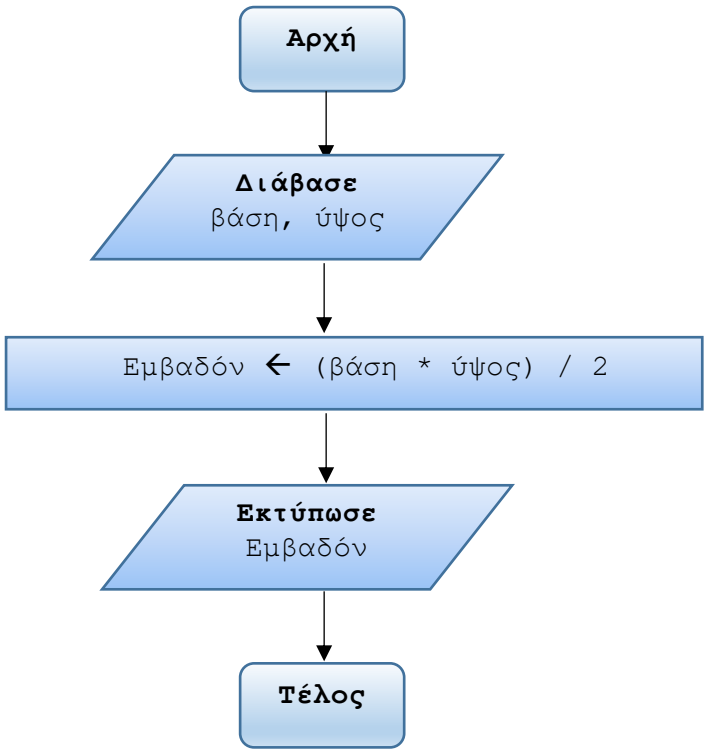
```
Αλγόριθμος Εμβαδόν_τριγώνου
    Διάβασε βάση
    Διάβασε ύψος
    Εμβαδόν ← (βάση * ύψος) / 2
    Εκτύπωσε Εμβαδόν
Τέλος Εμβαδόν_τριγώνου
```

Η λέξη **Αλγόριθμος** και **Τέλος** είναι **δηλωτικές εντολές** (δηλώνουν αντίστοιχα την αρχή και το τέλος του αλγορίθμου) ενώ οι λέξεις **Διάβασε** και **Εκτύπωσε** είναι **εκτελεστέες εντολές** (γίνεται κάποια ενέργεια). Εκτελεστέα εντολή είναι και η

```
Εμβαδόν ← (βάση * ύψος) / 2
```

καθώς είναι **εντολή εκχώρησης**. Η έκφραση στο δεξί μέρος αποτιμάται και το αποτέλεσμά της εκχωρείται στην μεταβλητή `Εμβαδόν` στο αριστερό μέρος.

Το αντίστοιχο διάγραμμα ροής είναι:

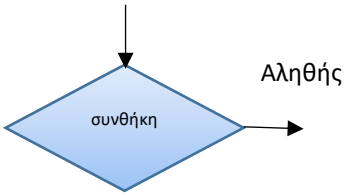
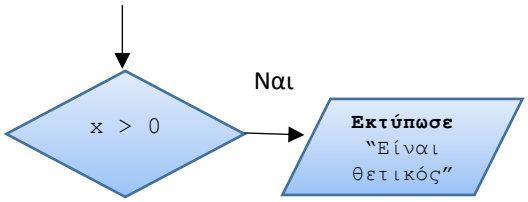
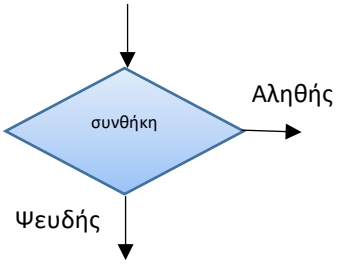
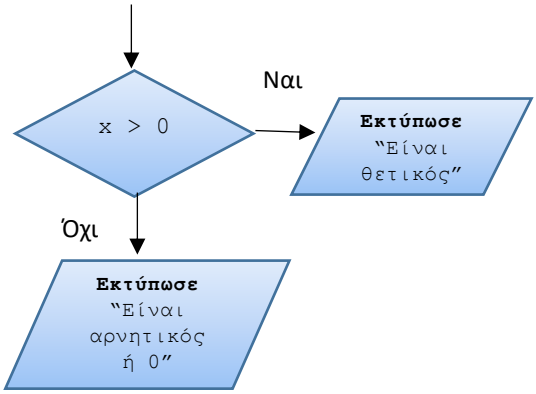


Όπως φαίνεται και από τα βέλη, η ροή εκτέλεσης του αλγορίθμου είναι ακολουθιακή, χωρίς κάποια διακλάδωση.

Δομή Επιλογής

Για προβλήματα όπου **απαιτείται να ληφθούν μία ή δύο αποφάσεις ανάλογα με την τιμή μίας έκφρασης**, χρησιμοποιούμε τη δομή επιλογής. Έχει **δύο μορφές**:

Ονομασία	Μορφή	Παράδειγμα:
Απλή Επιλογή *(μία απόφαση)*	**Αν** <συνθήκη> **τότε** *Εντολές* **Τέλος_αν**	**Αν** x > 0 **τότε** **Εκτύπωσε** "Είναι θετικός" **Τέλος_αν**

		
Σύνθετη επιλογή *(δύο αποφάσεις)*	**Αν** <συνθήκη> **τότε** *Εντολές1* **αλλιώς** *Εντολές2* **Τέλος_αν**	**Αν** x > 0 **τότε** **Εκτύπωσε** "Είναι θετικός" **αλλιώς** **Εκτύπωσε** "Είναι αρνητικός ή 0" **Τέλος_αν**
		

- Στη δομή της **Απλής Επιλογής**, μόνο αν η συνθήκη είναι αληθής θα εκτελεστεί η εντολή ή οι εντολές. Δεν προβλέπεται κάτι αν η συνθήκη είναι ψευδής.

- Στη δομή της **Σύνθετης Επιλογής**, αν η συνθήκη είναι αληθής τότε θα εκτελεστεί η πρώτη ομάδα εντολών ενώ αν η συνθήκη είναι ψευδής θα εκτελεστεί η δεύτερη ομάδα εντολών.

*Η **συνθήκη είναι μία λογική έκφραση** που αποτιμάται σε Αληθής ή Ψευδής (Ναι/Όχι, Ισχύει/Δεν Ισχύει).*

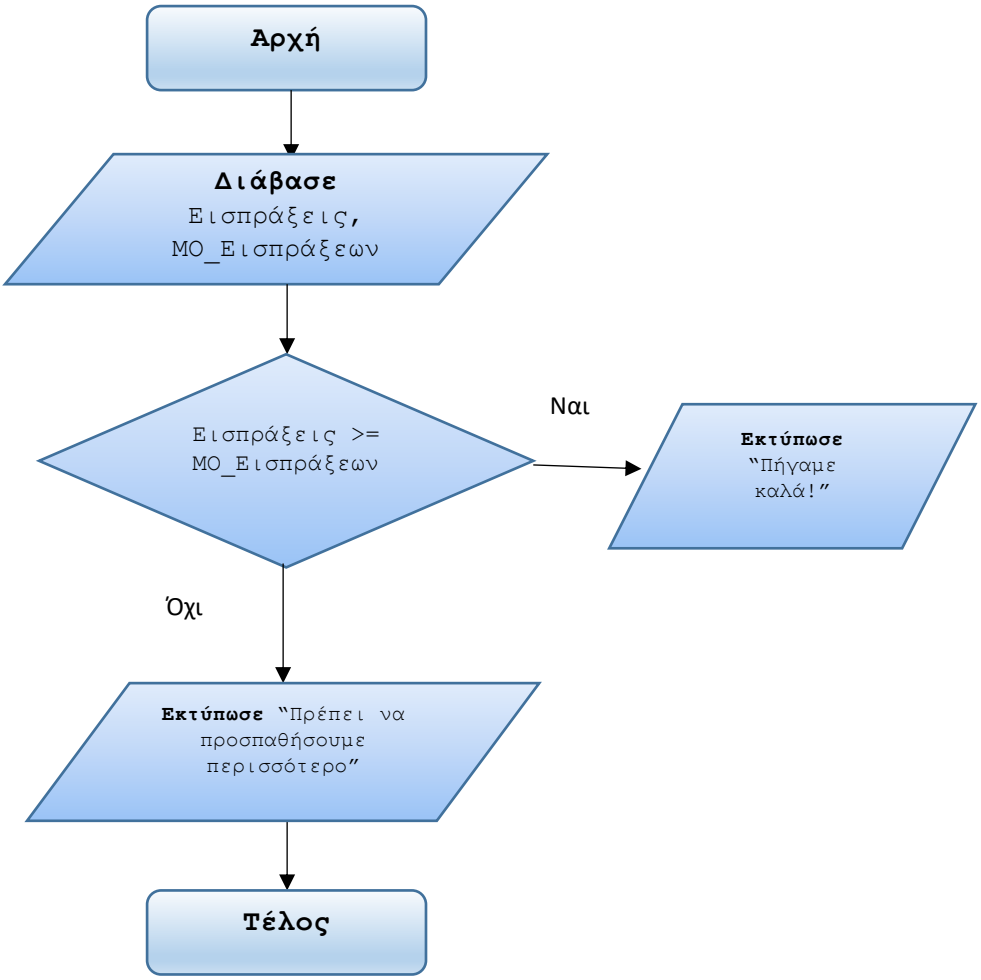
Παράδειγμα: Να διαβασθούν οι *εισπράξεις* που έκανε μία ποδοσφαιρική ομάδα κατά τη διάρκεια της σεζόν καθώς και ο *μέσος όρος των εισπράξεων* όλων των ποδοσφαιρικών ομάδων της κατηγορίας. Να εκτυπωθεί κατάλληλο μήνυμα αν οι εισπράξεις ήταν ίσες ή μεγαλύτερες από το μέσο όρο και άλλο μήνυμα αν ήταν μικρότερες.

Τα δεδομένα είναι οι εισπράξεις και ο μέσος όρος τους. Το ζητούμενο είναι η εκτύπωση κατάλληλου μηνύματος ανάλογα με το αποτέλεσμα της σύγκρισής τους.

Με ψευδογλώσσα:

```
Αλγόριθμος Εισπράξεις_ομάδας
    Διάβασε Εισπράξεις, ΜΟ_Εισπράξεων
    Αν Εισπράξεις >= ΜΟ_Εισπράξεων τότε
      Εκτύπωσε "Πήγαμε καλά!"
    αλλιώς
      Εκτύπωσε "Πρέπει να προσπαθήσουμε περισσότερο"
    Τέλος_αν
Τέλος Εισπράξεις_ομάδας
```

Με διάγραμμα ροής:



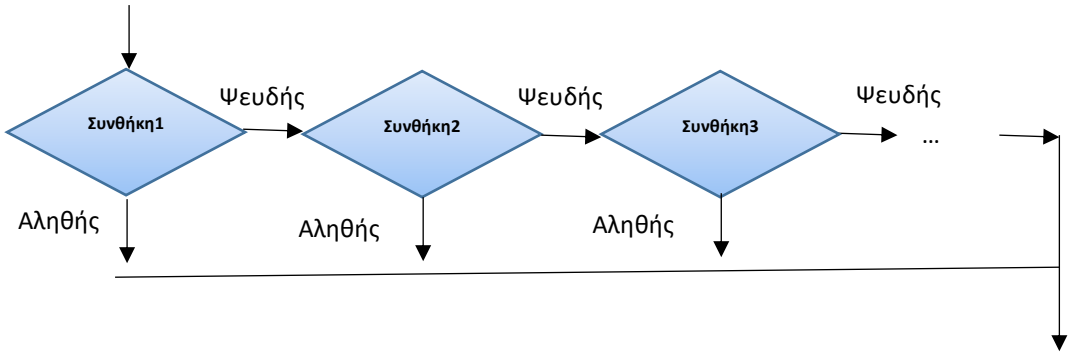
Δομή Πολλαπλής Επιλογής

Για προβλήματα που **απαιτείται να ληφθούν περισσότερες από δύο αποφάσεις ανάλογα πάντα με την τιμή μίας έκφρασης** τότε χρησιμοποιούμε τη δομή της πολλαπλής επιλογής.

Έχει **δύο μορφές**. *Αν οι αποφάσεις (επιλογές) είναι λίγες* μπορούμε να χρησιμοποιήσουμε το παρακάτω σχήμα:

```
Αν <συνθήκη1> τότε Εντολές1
   αλλιώς_αν <συνθήκη2> τότε Εντολές2
   αλλιώς_αν <συνθήκη3> τότε Εντολές3
   …
   αλλιώς Εντολές_άλλες
Τέλος_αν
```

Σε αντίστοιχο λογικό διάγραμμα :



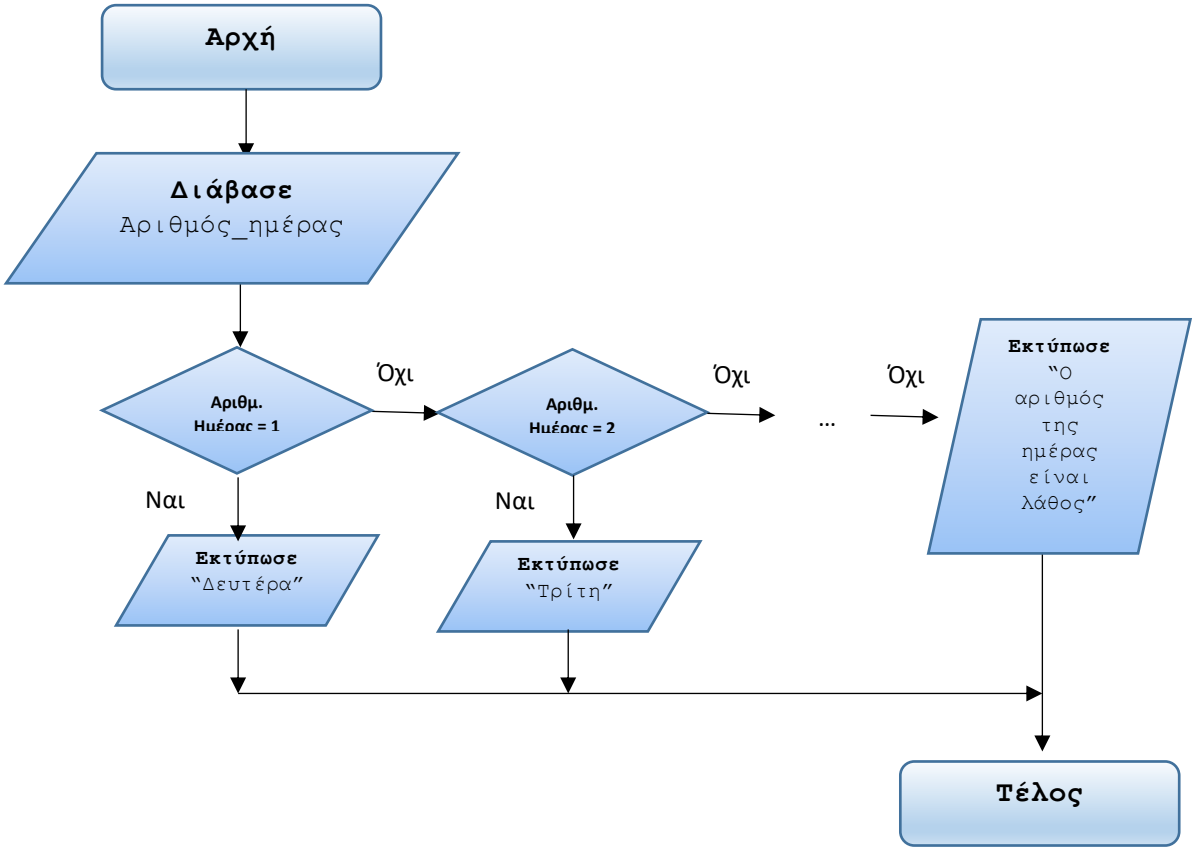
Παράδειγμα: Γράψτε έναν αλγόριθμο που διαβάζει τον αριθμό της ημέρας και τυπώνει την αντίστοιχη περιγραφή. Δηλαδή, αν διαβάσει το 1 να τυπώσει "Δευτέρα", αν διαβάσει το 2 να τυπώσει "Τρίτη" κ.ο.κ.

Σε ψευδογλώσσα:

```
Αλγόριθμος Περιγραφή_ημέρας
    Διάβασε Αριθμός_ημέρας
    Αν Αριθμός_ημέρας = 1 τότε Εκτύπωσε "Δευτέρα"
       αλλιώς_αν  Αριθμός_ημέρας = 2 τότε Εκτύπωσε "Τρίτη"
       αλλιώς_αν  Αριθμός_ημέρας = 3 τότε Εκτύπωσε "Τετάρτη"
       αλλιώς_αν  Αριθμός_ημέρας = 4 τότε Εκτύπωσε "Πέμπτη"
       αλλιώς_αν  Αριθμός_ημέρας = 5 τότε Εκτύπωσε "Παρασκευή"
       αλλιώς_αν  Αριθμός_ημέρας = 6 τότε Εκτύπωσε "Σάββατο"
       αλλιώς_αν  Αριθμός_ημέρας = 7 τότε Εκτύπωσε "Κυριακή"
       αλλιώς  Εκτύπωσε "Ο αριθμός της ημέρας είναι λάθος!"
    Τέλος_αν
Τέλος Περιγραφή_ημέρας
```

Παρατηρούμε ότι σε περίπτωση που ο αριθμός της ημέρας δεν είναι μεταξύ 1 και 7 τότε εκτελείται η εντολή στο τμήμα `αλλιώς`. Με άλλα λόγια, **αν οποιαδήποτε από τις παραπάνω συνθήκες δεν αποτιμηθεί σε `Αληθής` τότε εκτελείται η ομάδα εντολών στο τμήμα `αλλιώς`** (*που πάντα μπαίνει τελευταίο σε αυτό το σχήμα της πολλαπλής επιλογής*).

Το αντίστοιχο λογικό διάγραμμα (*του οποίου έχουμε παραλείψει κάποια τμήματα που είναι παρόμοια λόγω χώρου*) :



Όταν οι διαφορετικές επιλογές είναι πολλές (όπως στο παραπάνω παράδειγμα) είναι προτιμότερο να ακολουθήσουμε το **δεύτερο σχήμα της πολλαπλής επιλογής**:

```
Επίλεξε <έκφραση>
    Περίπτωση τιμή_έκφρασης_1
    Περίπτωση τιμή_έκφρασης_2
     ...
    Περίπτωση αλλιώς τιμή_έκφρασης_αλλιώς
Τέλος_επιλογών
```

Σε πολλές περιπτώσεις, η έκφραση μπορεί να είναι μόνο μία μεταβλητή, την οποία ελέγχουμε ως προς κάποιες διακριτές διαφορετικές τιμές ή αν βρίσκεται μέσα σε ένα εύρος τιμών.

Το αντίστοιχο παράδειγμα αλγορίθμου, που είδαμε παραπάνω, γράφεται ισοδύναμα:

```
Αλγόριθμος Περιγραφή_ημέρας
    Διάβασε Αριθμός_ημέρας
    Επίλεξε Αριθμός_ημέρας
        Περίπτωση 1
            Εκτύπωσε "Δευτέρα"
        Περίπτωση 2
            Εκτύπωσε "Τρίτη"
        Περίπτωση 3
            Εκτύπωσε "Τετάρτη"
        Περίπτωση 4
            Εκτύπωσε "Πέμπτη"
        Περίπτωση 5
            Εκτύπωσε "Παρασκευή"
        Περίπτωση 6
            Εκτύπωσε "Σάββατο"
        Περίπτωση 7
            Εκτύπωσε "Κυριακή"
        Περίπτωση αλλιώς
            Εκτύπωσε "Ο αριθμός της ημέρας είναι λάθος!"
    Τέλος_επιλογών
Τέλος Περιγραφή_ημέρας
```

Είναι εμφανές πόσο πιο ευανάγνωστος και εύληπτος είναι ο αλγόριθμος στο σχήμα `Επίλεξε…Τέλος_επιλογών`. Να σημειώσουμε ότι το λογικό διάγραμμα δεν αλλάζει.

Τα δύο σχήματα της πολλαπλής επιλογής είναι ισοδύναμα από άποψη αποτελεσματικότητας. Μπορούμε πάντα να μετατρέπουμε τον αλγόριθμο από το ένα σχήμα στο άλλο.

Παράδειγμα: Γράψτε έναν αλγόριθμο που διαβάζει τον μέσο ετήσιο βαθμό ενός μαθητή (από το 1 μέχρι το 20) και τυπώνει την αντίστοιχο χαρακτηρισμό του. Για παράδειγμα, αν διαβάσει το 11,5 να τυπώσει τον χαρακτηρισμό "Μέτρια" ενώ αν διαβάσει το 18,8 να τυπώσει το "Άριστα". Να βγάλει και κατάλληλο μήνυμα όταν ο βαθμός που δόθηκε δεν είναι μεταξύ 1 και 20.

Θα χρειαστούμε μία μεταβλητή για να αποθηκευτεί ο βαθμός που θα διαβαστεί. Προφανώς, θα ελέγξουμε την τιμή της για συγκεκριμένο εύρος τιμών: 1-9,4 (κακώς) 9,5-12,4 (μέτρια) 12,5-15,4 (καλά) 15,5-18,4 (πολύ καλά) και 18,5-20 (άριστα).

- Με την **1η μορφή της πολλαπλής επιλογής**:

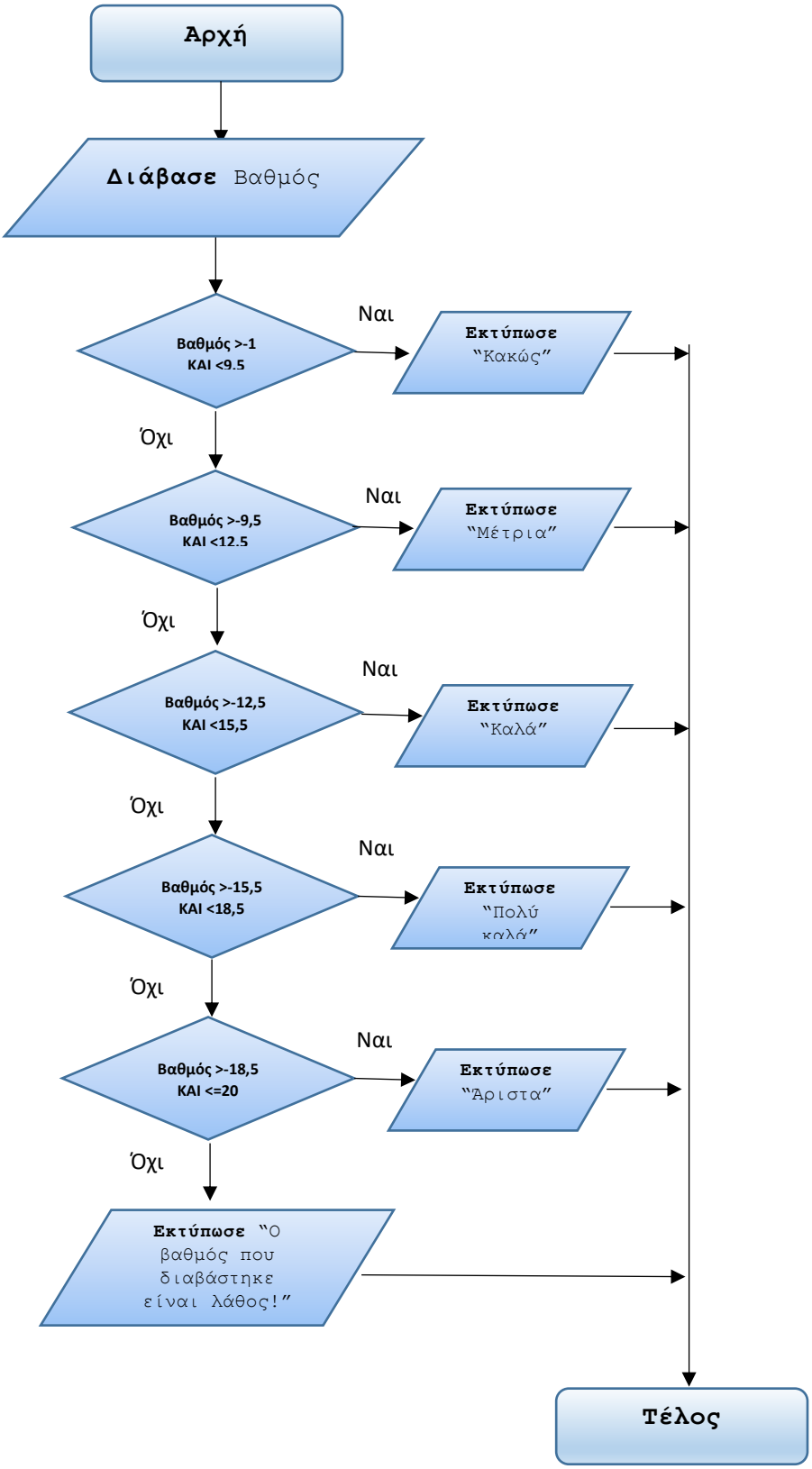
```
Αλγόριθμος Χαρακτηρισμός_Βαθμού
    Διάβασε Βαθμός
    Αν Βαθμός >= 1 ΚΑΙ Βαθμός < 9,5 τότε Εκτύπωσε "Κακώς"
       αλλιώς_αν  Βαθμός >= 9,5 ΚΑΙ Βαθμός <12,5 τότε Εκτύπωσε
                                                    "Μέτρια"
       αλλιώς_αν  Βαθμός >= 12,5 ΚΑΙ Βαθμός <15,5 τότε
                                          Εκτύπωσε "Καλά"
       αλλιώς_αν  Βαθμός >= 15,5 ΚΑΙ Βαθμός <18,5  τότε
                                       Εκτύπωσε "Πολύ καλά"
       αλλιώς_αν  Βαθμός >= 18,5 ΚΑΙ Βαθμός <=20 τότε Εκτύπωσε
                                                    "Άριστα"
       αλλιώς  Εκτύπωσε "Ο βαθμός που διαβάστηκε είναι λάθος!"
    Τέλος_αν
Τέλος Χαρακτηρισμός_Βαθμού
```

- Με την **2η μορφή της πολλαπλής επιλογής:**

```
Αλγόριθμος Χαρακτηρισμός_Βαθμού
    Διάβασε Βαθμός
    Επίλεξε Βαθμός
        Περίπτωση >=1 ΚΑΙ <9,5
            Εκτύπωσε "Κακώς"
        Περίπτωση >=9,5 ΚΑΙ <12,5
```

```
            Εκτύπωσε "Μέτρια"
        Περίπτωση >=12,5 ΚΑΙ <15,5
            Εκτύπωσε "Καλά"
        Περίπτωση >=15,5 ΚΑΙ <18,5
            Εκτύπωσε "Πολύ καλά"
        Περίπτωση >=18,5 ΚΑΙ <=20
            Εκτύπωσε "Άριστα"
        Περίπτωση αλλιώς
            Εκτύπωσε "Ο βαθμός που διαβάστηκε είναι λάθος!"
    Τέλος_επιλογών
Τέλος Χαρακτηρισμός_Βαθμού
```

Το αντίστοιχο λογικό διάγραμμα:


Αρχή
Διάβασε Βαθμός
Βαθμός >-1 ΚΑΙ <9.5
Ναι
Εκτύπωσε "Κακώς"
Όχι
Βαθμός >-9,5 ΚΑΙ <12.5
Ναι
Εκτύπωσε "Μέτρια"
Όχι
Βαθμός >-12,5 ΚΑΙ <15,5
Ναι
Εκτύπωσε "Καλά"
Όχι
Βαθμός >-15,5 ΚΑΙ <18,5
Ναι
Εκτύπωσε "Πολύ καλά"
Όχι
Βαθμός >-18,5 ΚΑΙ <=20
Ναι
Εκτύπωσε "Άριστα"
Όχι
Εκτύπωσε "Ο βαθμός που διαβάστηκε είναι λάθος!"
Τέλος

Δεν είναι υποχρεωτικό να θέτουμε το τμήμα αλλιώς *στα δύο σχήματα της δομής πολλαπλής επιλογής. Η χρήση του ή όχι εξαρτάται από τις απαιτήσεις του προβλήματος.*

Εμφωλευμένες διαδικασίες

Πρόκειται για *ειδικές περιπτώσεις πολλαπλών επιλογών*, όπου μία δομή επιλογής βρίσκεται μέσα σε μία άλλη δομή επιλογής (*εμφωλευμένη δομή*). Ένα γενικό παράδειγμα είναι το εξής:

```
Αν <συνθήκη> τότε
```

```
Αν <συνθήκη> τότε
   Εντολές1
αλλιώς
   Εντολές2
Τέλος_αν
```

Εσώτερη δομή
Αν..τότε...αλλιώς.
Εκτελείται μόνο όταν η **συνθήκη** της εξώτερης δομής Αν..τότε..αλλιώς είναι **Αληθής**.

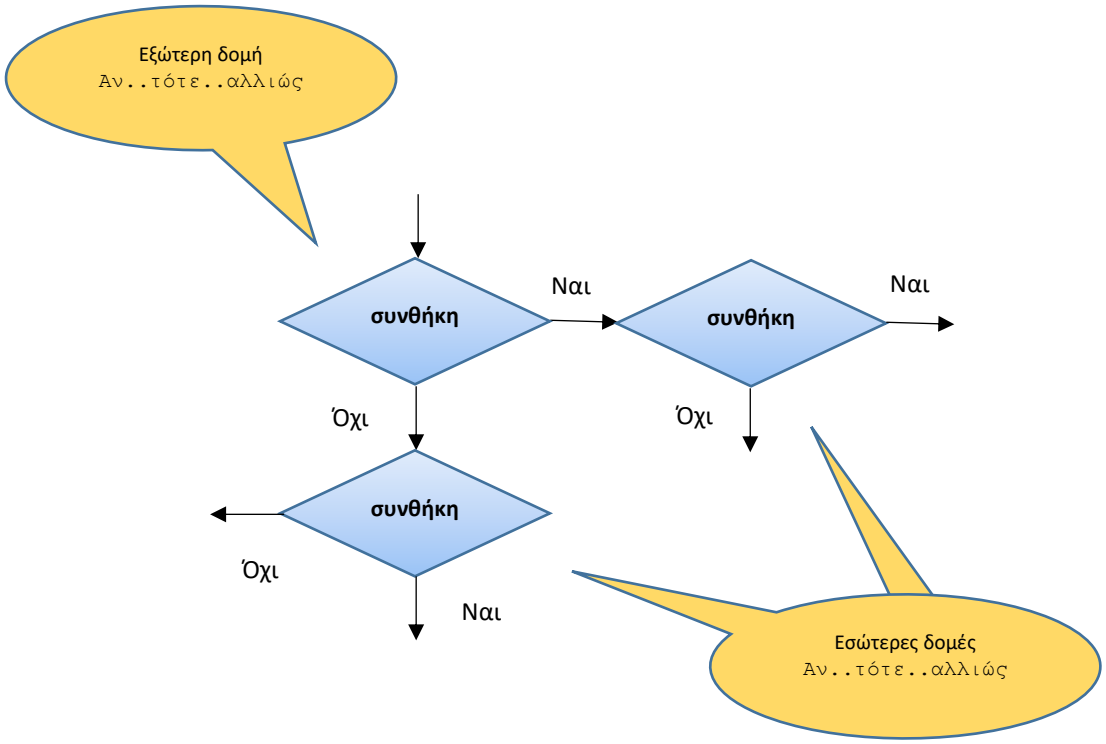
```
αλλιώς
```

```
Αν <συνθήκη> τότε
   Εντολές3
αλλιώς
   Εντολές4
Τέλος_αν
```

Εσώτερη δομή
Αν..τότε...αλλιώς.
Εκτελείται μόνο όταν η **συνθήκη** της εξώτερης δομής Αν..τότε..αλλιώς είναι **Ψευδής**.

```
Τέλος_αν
```

Το αντίστοιχο **λογικό διάγραμμα μίας εμφωλευμένης διαδικασίας** γενικά είναι το εξής:



Φυσικά, **οι δομές επιλογής μπορεί να είναι απλές ή σύνθετες**. Σε κάθε περίπτωση, δεν πρέπει να ξεχνάμε ότι αυτές οι εμφωλευμένες διαδικασίες αποτελούν εξειδικευμένα σχήματα πολλαπλής επιλογής.

Παράδειγμα: Γράψτε έναν αλγόριθμο που διαβάζει τον μέσο ετήσιο βαθμό ενός μαθητή (από το 1 μέχρι το 20) και τη Διαγωγή του («Εξαιρετική», «Καλή», «Μεμπτή»). Αν ο βαθμός του είναι πάνω από 15,5 και η διαγωγή του είναι εξαιρετική ή καλή να εκτυπώνει το μήνυμα «Πολύ καλός μαθητής με καλή διαγωγή» αλλιώς να εκτυπώνει το μήνυμα «Πολύ καλός μαθητής αλλά με προβληματική διαγωγή». Αν ο βαθμός του είναι μεταξύ 12,5 και 15,4 και η διαγωγή του εξαιρετική ή καλή να εκτυπώνει το μήνυμα «Καλός μαθητής με καλή διαγωγή» αλλιώς να εκτυπώνει το μήνυμα «Καλός μαθητής αλλά με προβληματική διαγωγή».

Φυσικά, θα μπορούσαμε να έχουμε κι άλλες περιπτώσεις αλλά είναι εμφανές ότι είναι ένα πρόβλημα που επιλύεται με αλγόριθμο πολλαπλής επιλογής.

Ένας προτεινόμενος αλγόριθμος σε ψευδογλώσσα είναι ο ακόλουθος:

Πρόταση 1 (με εμφωλευμένη διαδικασία):

```
Αλγόριθμος Χαρακτηρισμός_μαθητή
    Διάβασε Βαθμός, Διαγωγή
    Αν Βαθμός >= 15,5 τότε
        Αν Διαγωγή = "Εξαιρετική" Ή Διαγωγή = "Καλή" τότε
            Εκτύπωσε "Πολύ καλός μαθητής με καλή διαγωγή"
        αλλιώς
            Εκτύπωσε "Πολύ καλός μαθητής αλλά με προβληματική διαγωγή"
    αλλιώς_αν Βαθμός >= 12,5 ΚΑΙ Βαθμός < 15,5 τότε
        Αν Διαγωγή = "Εξαιρετική" Ή Διαγωγή = "Καλή" τότε
            Εκτύπωσε "Καλός μαθητής με καλή διαγωγή"
        αλλιώς
            Εκτύπωσε "Καλός μαθητής αλλά με προβληματική διαγωγή"
    Τέλος_αν
Τέλος Χαρακτηρισμός_μαθητή
```

*Είναι σημαντικό να προσέχουμε τη **στοίχιση των αλγοριθμικών δομών**. Για παράδειγμα, οι λέξεις* `Αν`, `αλλιώς` *και* `Τέλος_αν` *είναι στην ίδια κατακόρυφη ευθεία. Ενδεχόμενες εσώτερες δομές επιλογής να βρίσκονται στην δική τους ευθεία. Το ίδιο συμβαίνει και στις επαναληπτικές δομές που θα δούμε παρακάτω.* **Με αυτόν τον τρόπο ο αλγόριθμος γίνεται πιο οργανωμένος και εύληπτος.**

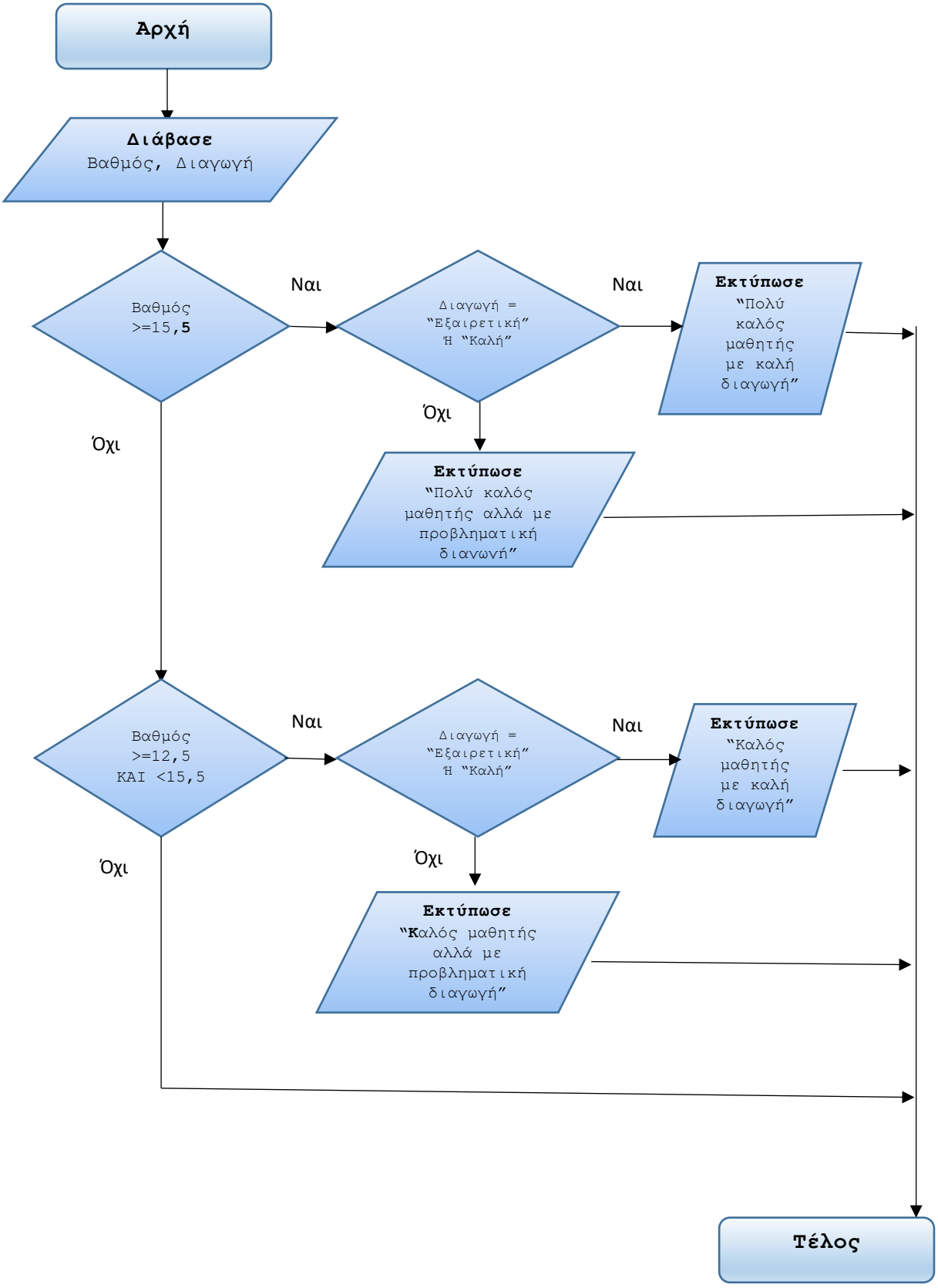
Μία άλλη πρόταση είναι *να απομακρύνουμε την εσώτερη δομή επιλογής* και να κάνουμε *πιο σύνθετες συνθήκες ελέγχου*:

Πρόταση 2 (με δομή πολλαπλής επιλογής με σύνθετες λογικές εκφράσεις):

```
Αλγόριθμος Χαρακτηρισμός_μαθητή
    Διάβασε Βαθμός, Διαγωγή
    Αν Βαθμός >= 15,5 ΚΑΙ (Διαγωγή = "Εξαιρετική" Ή Διαγωγή =
                                                "Καλή") τότε
        Εκτύπωσε "Πολύ καλός μαθητής με καλή διαγωγή"
    αλλιώς_αν Βαθμός >= 15,5 ΚΑΙ Διαγωγή = "Μεμπτή" τότε
        Εκτύπωσε "Πολύ καλός μαθητής αλλά με προβληματική διαγωγή"
    αλλιώς_αν Βαθμός >= 12,5 ΚΑΙ Βαθμός < 15,5 ΚΑΙ
            (Διαγωγή = "Εξαιρετική" Ή Διαγωγή = "Καλή") τότε
        Εκτύπωσε "Καλός μαθητής με καλή διαγωγή"
    αλλιώς_αν Βαθμός >= 12,5 ΚΑΙ Βαθμός < 15,5 ΚΑΙ
                                        Διαγωγή = "Μεμπτή" τότε
        Εκτύπωσε "Καλός μαθητής αλλά με προβληματική διαγωγή"
    Τέλος_αν
Τέλος Χαρακτηρισμός_μαθητή
```

Το τί θα διαλέξει κάποιος εξαρτάται από το προσωπικό του στυλ. Πιθανόν η 1η πρόταση να είναι περισσότερο κατανοητή αλλά και πάλι το βάθος των εμφωλεύσεων μπορεί να είναι μεγάλο ανάλογα με το μέγεθος του προβλήματος και τα κριτήρια επιλογών.

Το αντίστοιχο λογικό διάγραμμα της 1ης πρότασης είναι το εξής:


Αρχή
Διάβασε
Βαθμός, Διαγωγή
Βαθμός >=15,5
Ναι
Διαγωγή = "Εξαιρετική" Ή "Καλή"
Ναι
Εκτύπωσε "Πολύ καλός μαθητής με καλή διαγωγή"
Όχι
Εκτύπωσε "Πολύ καλός μαθητής αλλά με προβληματική διανωγή"
Όχι
Βαθμός >=12,5 ΚΑΙ <15,5
Ναι
Διαγωγή = "Εξαιρετική" Ή "Καλή"
Ναι
Εκτύπωσε "Καλός μαθητής με καλή διαγωγή"
Όχι
Εκτύπωσε "Καλός μαθητής αλλά με προβληματική διαγωγή"
Τέλος

Λογικές πράξεις

Οι λογικές πράξεις εφαρμόζονται επί λογικών εκφράσεων και είναι οι εξής:

Λογική πράξη	Τελεστής	Παράδειγμα
Σύζευξη	**ΚΑΙ** (αγγλικό AND)	`X >= 1 ΚΑΙ X <= 10`
Διάζευξη	**Ή** (αγγλικό OR)	`X < 1 Ή X > 10`
Άρνηση	**ΟΧΙ** (αγγλικό NOT)	`ΟΧΙ (X >= 10)`

Στον παρακάτω πίνακα, έχουμε δύο λογικές εκφράσεις Α και Β και παρατηρούμε το αποτέλεσμα που δίνει κάθε πράξη μεταξύ τους:

Έκφραση Α	Έκφραση Β	Α **ΚΑΙ** Β	Α **Ή** Β	**ΟΧΙ** Α
Αληθής	Αληθής	Αληθής	Αληθής	Ψευδής
Αληθής	Ψευδής	Ψευδής	Αληθής	Ψευδής
Ψευδής	Αληθής	Ψευδής	Αληθής	Αληθής
Ψευδής	Ψευδής	Ψευδής	Ψευδής	Αληθής

Βλέποντας τα αποτελέσματα του παραπάνω πίνακα παρατηρούμε τα εξής:

- Στην πράξη **ΚΑΙ** όταν η μία πρόταση είναι Ψευδής τότε το αποτέλεσμα είναι Ψευδής.
- Στην πράξη **Ή** όταν η μία πρόταση είναι Αληθής τότε το αποτέλεσμα είναι Αληθής.
- Στην πράξη **ΟΧΙ** γίνεται αντιστροφή της τιμής.

Ιεραρχία (προτεραιότητα) λογικών τελεστών

Σε περιπτώσεις πολύπλοκων λογικών πράξεων (όπου εμπλέκονται πολλές εκφράσεις με λογικούς τελεστές **ΚΑΙ**, **Ή**, **ΟΧΙ**) για την σωστή αποτίμηση της τελικής τιμής πρέπει να έχουμε υπόψη ότι:

1. Οι πράξεις **ΟΧΙ** προηγούνται.
2. Οι πράξεις **ΚΑΙ** ακολουθούν.
3. Οι πράξεις **Ή** ακολουθούν τελευταίες.

Μερικές γενικές παρατηρήσεις:

- Οι λογικές πράξεις, συνήθως, εφαρμόζονται στις συνθήκες ελέγχου μίας δομής επιλογής ή πολλαπλής επιλογής (*καθώς και στις συνθήκες ελέγχου μίας επαναληπτικής δομής, όπως θα δούμε παρακάτω*).

- Οσοδήποτε πολύπλοκη και να είναι μία λογική πράξη, η τελική αποτίμησή της είναι ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ. Γι' αυτό άλλωστε χρησιμοποιούνται εκτενώς και στις συνθήκες ελέγχου.

- Όπως και στις αριθμητικές πράξεις, και στις λογικές πράξεις μπορούμε να αλλάξουμε την προτεραιότητα με παρενθέσεις. Έτσι, αν θέλουμε μία λογική πράξη **Ή** να προηγηθεί μίας **ΚΑΙ** τότε την βάζουμε εντός παρενθέσεων.
  Παράδειγμα:

`ΟΧΙ Χ >= 1 ΚΑΙ Χ <= 10`	Πρώτα θα αποτιμηθεί η `ΟΧΙ Χ >= 1` και κατόπιν θα γίνει **ΚΑΙ** με την `Χ <= 10`.
`ΟΧΙ (Χ >= 1 ΚΑΙ Χ <= 10)`	Πρώτα θα αποτιμηθεί η έκφραση `Χ >= 1 ΚΑΙ Χ <= 10` και κατόπιν θα γίνει αντιστροφή με την **ΟΧΙ**.

  Δοκιμάστε μόνοι σας να ελέγξετε τις δύο περιπτώσεις για διάφορες τιμές της `Χ`.

Δομή επανάληψης

Πολλά προβλήματα για να επιλυθούν απαιτούν κατάλληλες επαναληπτικές διαδικασίες. Σε γενικές γραμμές, μία δομή επανάληψης μας επιτρέπει να εκτελέσουμε μία ομάδα εντολών πολλές φορές.

Έχει 3 μορφές:

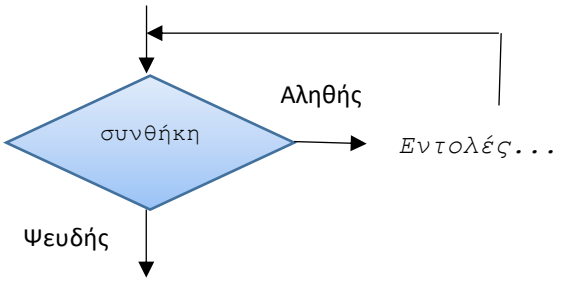
- **Όσο..επανάλαβε**
- **Αρχή_επανάληψης..Μέχρις_ότου**
- **Για..από..μέχρι**

Κάθε μία δομή επανάληψης από τις τρεις παραπάνω, είναι κατάλληλη όταν η επαναληπτικότητα της διαδικασίας έχει κάποιες συγκεκριμένες ιδιαιτερότητες. Για παράδειγμα, όταν δεν γνωρίζουμε πόσες φορές θα επαναληφθεί μία ομάδα εντολών θα χρησιμοποιήσουμε μία από τις δύο πρώτες (και με κανέναν τρόπο την τρίτη). Ή το αντίθετο, όταν γνωρίζουμε τον αριθμό των επαναλήψεων τότε η καταλληλότερη είναι η τρίτη.

Παρακάτω, εξετάζουμε κάθε μία ξεχωριστά.

Όσο..επανάλαβε

Μορφή:

Όσο <συνθήκη> **επανάλαβε** Εντολές… **Τέλος_επανάληψης**	

Γενικά: Όσο η συνθήκη είναι αληθής τότε επαναλαμβάνονται οι εντολές μέσα στο σώμα της δομής επανάληψης `Όσο..επανάλαβε`.

Παρατηρήσεις:

- **Η συνθήκη, ως γνωστόν, είναι μία λογική έκφραση**, με αποτίμηση `ΑΛΗΘΗΣ/ΨΕΥΔΗΣ`. Μπορεί να είναι οσοδήποτε πολύπλοκη ακόμα και με λογικές πράξεις **`ΚΑΙ, Ή ΌΧΙ`**.

- **Πρώτα γίνεται έλεγχος της συνθήκης**. Αν είναι `ΑΛΗΘΗΣ`, τότε εκτελούνται οι εντολές στο εσωτερικό της.

- Αφού εκτελεστούν οι εντολές στο εσωτερικό της δομής επανάληψης, η ροή εκτέλεσης επιστρέφει στην αρχή και ξαναγίνεται έλεγχος της συνθήκης. Αν είναι `ΑΛΗΘΗΣ`, τότε εκτελούνται ξανά οι εντολές στο εσωτερικό της. Και αυτό γίνεται συνέχεια μέχρι η συνθήκη να αποτιμηθεί σε `ΨΕΥΔΗΣ`.

- Όταν η συνθήκη αποτιμηθεί σε `ΨΕΥΔΗΣ` τότε η ροή εκτέλεσης συνεχίζεται παρακάτω. ΠΡΕΠΕΙ κάποια στιγμή η συνθήκη να αποτιμηθεί σε `ΨΕΥΔΗΣ` αλλιώς η ροή εκτέλεσης θα εμπλακεί σε μία **ατέρμονη επανάληψη**.

- **Επειδή πρώτα γίνεται ο έλεγχος της συνθήκης, υπάρχει περίπτωση να αποτιμηθεί εξ΄ αρχής ΨΕΥΔΗΣ και συνεπώς να μην εκτελεστούν ποτέ οι εντολές στο εσωτερικό της δομής `Όσο..επανάλαβε`.**

- Μία **δομή επανάληψης** αποκαλείται στην ορολογία της Πληροφορικής **βρόχος (loop).**

- **Πολλές φορές, στις δομές επανάληψης χρησιμοποιούμε ειδικές μεταβλητές, που ονομάζονται μετρητές (***συνήθως η ονομασία τους είναι `i` ή `j` κλπ.),* οι οποίοι καθορίζουν τον αριθμό των επαναλήψεων.

Παράδειγμα 1: Να εκτυπωθεί 5 φορές το μήνυμα “Ανάπτυξη εφαρμογών”.

Χωρίς δομή επανάληψης μπορούμε να κατασκευάσουμε τον αλγόριθμο ως εξής:

```
Αλγόριθμος Πολλαπλή_εκτύπωση_μηνύματος
    Εκτύπωσε "Ανάπτυξη εφαρμογών"
    Εκτύπωσε "Ανάπτυξη εφαρμογών"
    Εκτύπωσε "Ανάπτυξη εφαρμογών"
```

```
        Εκτύπωσε "Ανάπτυξη εφαρμογών"
        Εκτύπωσε "Ανάπτυξη εφαρμογών"
Τέλος Πολλαπλή_εκτύπωση_μηνύματος
```

Είναι προφανές ότι αν η ίδια εντολή πρέπει να εκτελεστεί 100 φορές κάτι τέτοιο θα είναι ασύμφορο.

Με τη δομή επανάληψης Όσο..επανάλαβε, ο αλγόριθμος διαμορφώνεται ως εξής:

```
Αλγόριθμος Πολλαπλή_εκτύπωση_μηνύματος
    i ← 1
    Όσο i <= 5 επανάλαβε
        Εκτύπωσε "Ανάπτυξη εφαρμογών"
        i ← i + 1
    Τέλος_επανάληψης
Τέλος Πολλαπλή_εκτύπωση_μηνύματος
```

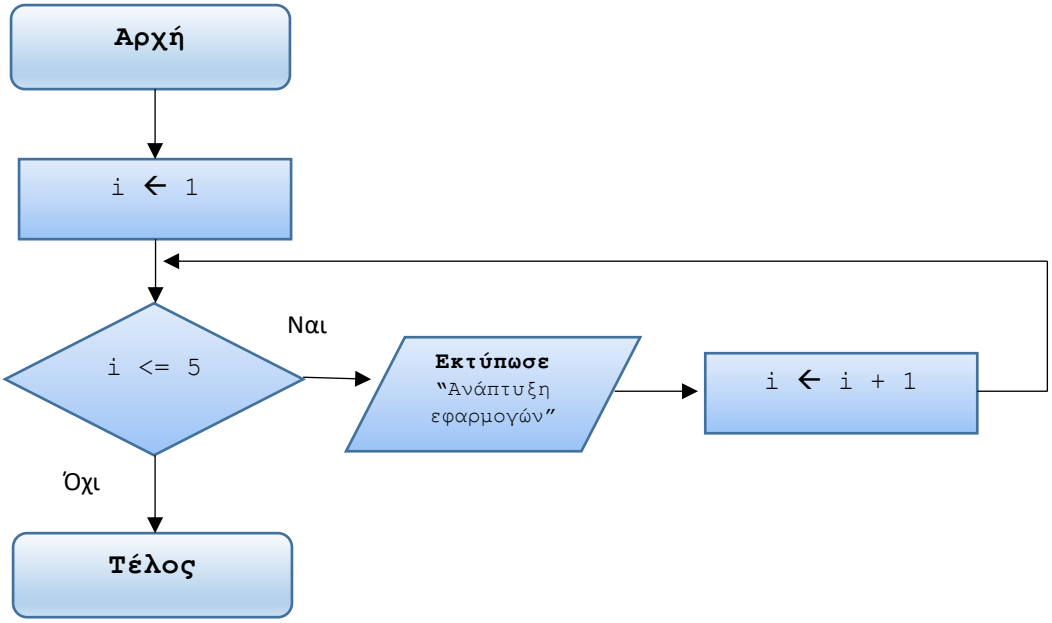
Η μεταβλητή i είναι ο μετρητής. Μέσω αυτού ελέγχουμε αν έχουμε ξεπεράσει το πλήθος των επαναλήψεων (5).

Κρίσιμη είναι η σημασία του **μετρητή i**.

- **Πριν την επανάληψη**, λαμβάνει μία **αρχική τιμή** (εδώ 1).
- Όσο η συνθήκη είναι Αληθής, δηλαδή η τιμή του μετρητή είναι κάτω ή ίσο με 5, τότε εκτελούνται οι εντολές στο εσωτερικό της δομής επανάληψης. Εκτυπώνει το μήνυμα και **ακριβώς πριν το τέλος αυξάνει την τιμή του κατά 1**.

*Αυτή η **κυκλική εκτέλεση εντολών** που χαρακτηρίζει τις δομές επανάληψης, ονομάζεται **βρόχος (loop)**. Στην περίπτωση της Όσο..επανάλαβε, η κυκλική εκτέλεση των εντολών σταματάει όταν η συνθήκη (λογική έκφραση) αποτιμηθεί σε Ψευδής,*

Το αντίστοιχο λογικό διάγραμμα είναι το εξής:



Η δομή `Όσο . . επανάλαβε` είναι περισσότερο χρήσιμη όταν δεν γνωρίζουμε τον αριθμό των επαναλήψεων. Ένα παράδειγμα είναι η **επαναληπτική είσοδος στοιχείων**. Δηλαδή, διαβάζει ο αλγόριθμος μία ακολουθία δεδομένων (π.χ. αριθμών) χωρίς να γνωρίζουμε εκ των προτέρων πόσα είναι αυτά τα δεδομένα. Πρέπει, βέβαια, να γνωρίζουμε πότε τερματίζεται η είσοδος των δεδομένων (π.χ. όταν ο αριθμός που διαβάστηκε είναι 0).

Παράδειγμα 2: Να γραφτεί αλγόριθμος που διαβάζει μία ακολουθία αριθμών που αφορούν ημερήσιες εισπράξεις ενός καταστήματος. Αφού διαβαστούν όλοι οι αριθμοί (*εισπράξεις*) θα πρέπει να υπολογίσει το σύνολο των εισπράξεων του καταστήματος. Το τέλος της εισόδου των αριθμών θα σηματοδοτείται μόλις διαβαστεί ο αριθμός -1.

Εδώ, θα πρέπει να διαβάζεται συνεχώς (δηλαδή επαναληπτικά) κάθε αριθμός. Η επανάληψη τερματίζεται όταν διαβαστεί ο αριθμός -1, πράγμα που μας δηλώνει και τη ***συνθήκη τερματισμού της επανάληψης***. Το σημείο, επίσης, που πρέπει να προσέξουμε είναι ότι πρέπει κάθε φορά που διαβάζουμε έναν αριθμό (είσπραξη ημέρας) να τον προσθέτουμε σε έναν συσσωρευτή. Ο **συσσωρευτής (ή αθροιστής) είναι μία μεταβλητή που κρατάει το σύνολο (άθροισμα) των αριθμών που διαβάστηκαν**. Πρέπει **να τεθεί αρχικά σε 0** και μόλις, ολοκληρωθεί η επαναληπτική διαδικασία θα μας δώσει το τελικό σύνολο των εισπράξεων.

Η λογική είναι η εξής:

- Βήμα 1. Διάβασε τον αριθμό (είσπραξη ημέρας).
- Βήμα 2. Πρόσθεσέ τον στον συσσωρευτή.
- Βήμα 3. Αν υπάρχουν άλλοι αριθμοί, συνέχισε από το Βήμα 1 αλλιώς σταμάτησε.
- Βήμα 4. Τύπωσε την τιμή του συσσωρευτή.

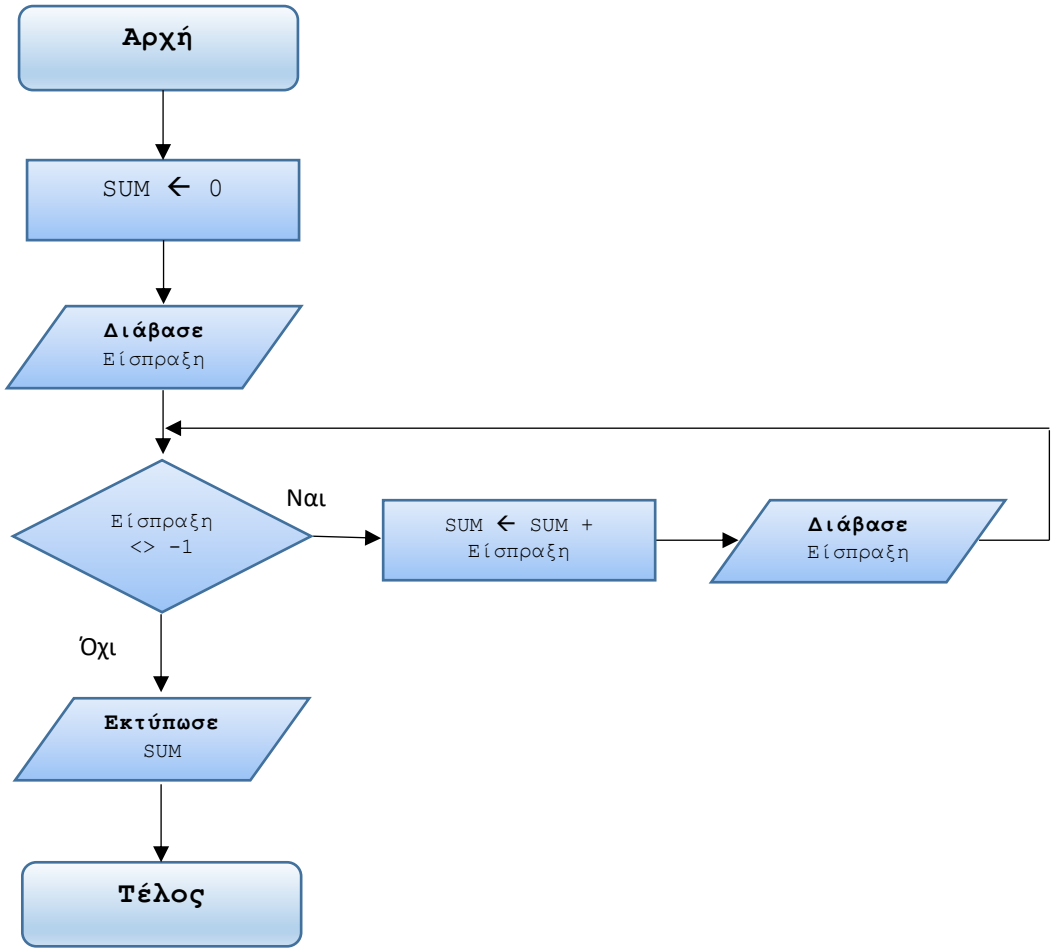
Ας δούμε τον αλγόριθμο σε ψευδογλώσσα:

```
Αλγόριθμος Επαναληπτική_είσοδος_στοιχείων
    SUM ← 0
    Διάβασε Είσπραξη
    Όσο Είσπραξη <> -1 επανάλαβε
        SUM ← SUM + Είσπραξη
        Διάβασε Είσπραξη
    Τέλος_επανάληψης
    Εκτύπωσε SUM
Τέλος Επαναληπτική_είσοδος_στοιχείων
```

Εδώ, δεν χρειάζεται μετρητής αφού δεν γνωρίζουμε πόσους αριθμούς θα διαβάσουμε (*αν και θα μπορούσε να αξιοποιηθεί σε άλλου είδους πρόβλημα π.χ., υπολογισμό μέσου όρου).* Ο πρώτος αριθμός διαβάζεται πριν την δομή επανάληψης. Στο εσωτερικό της δομής επανάληψης αξιοποιείται ο αριθμός που διαβάστηκε (εδώ προστίθεται σε έναν συσσωρευτή) και πριν το τέλος διαβάζεται ο επόμενος και επιστρέφει στη συνθήκη ελέγχου για την επόμενη επανάληψη. Αυτή είναι μία τυπική περίπτωση επαναληπτική εισόδου δεδομένων με την δομή Όσο…επανάλαβε.

Υπάρχει περίπτωση να μην εκτελεστεί καθόλου η επανάληψη; Ναι, αν ο πρώτος αριθμός που διαβάστηκε (πριν τη δομή Όσο.. επανάλαβε) είναι ο -1. Αυτό, βέβαια, υποδηλώνει ότι δεν δόθηκε κανένας αριθμός είσπραξης.

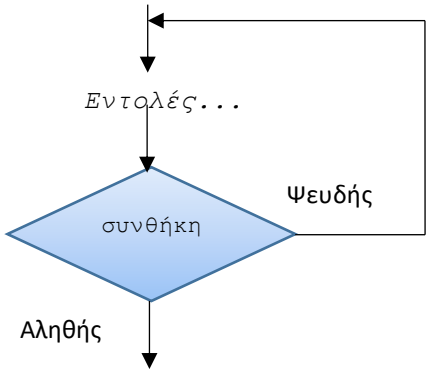
Το αντίστοιχο λογικό διάγραμμα είναι το εξής:



Αρχή_επανάληψης..Μέχρις_ότου

Πρόκειται για μία παρόμοια δομή επανάληψης με τη διαφορά ότι **οι εντολές εκτελούνται τουλάχιστον μία φορά**. Αυτό οφείλεται στη θέση της συνθήκης ελέγχου.

Μορφή:

Αρχή_επανάληψης Εντολές… **Μέχρις_ότου** <συνθήκη>	

Παρατηρήσεις:

- **Οι εντολές εκτελούνται συνεχώς όσο η συνθήκη είναι ΨΕΥΔΗΣ** (*αυτή είναι μία βασική διαφοροποίηση σε σχέση με την* `Όσο..επανάλαβε` *όπου οι εντολές εκτελούνται συνεχώς όσο η συνθήκη είναι* `ΑΛΗΘΗΣ`*).*

- Οι εντολές πρώτα εκτελούνται μία φορά και κατόπιν γίνεται ο έλεγχος της συνθήκης.

- Και αυτή η δομή επανάληψης είναι καταλληλότερη όταν δεν γνωρίζουμε εκ των προτέρων τον αριθμό των επαναλήψεων, όπως για παράδειγμα η **επαναληπτική είσοδος στοιχείων**.

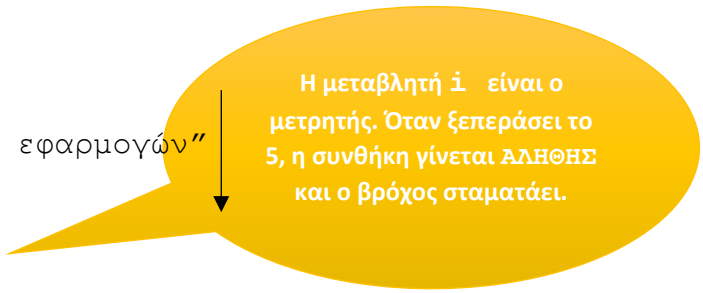
- Αν γνωρίζουμε ότι οι εντολές θα εκτελεστούν οπωσδήποτε μία φορά τότε είναι καταλληλότερη κι από την `Όσο..επανάλαβε`.

Ας δούμε ξανά τα δύο παραδείγματα της `Όσο..επανάλαβε` αλλά τώρα θα τα επιλύσουμε με την `Αρχή_επανάληψης..Μέχρις_ότου`. Έτσι, θα δούμε και τη διαφοροποίηση στην επίλυση.

Παράδειγμα 1: Να εκτυπωθεί 5 φορές το μήνυμα “Ανάπτυξη εφαρμογών”.

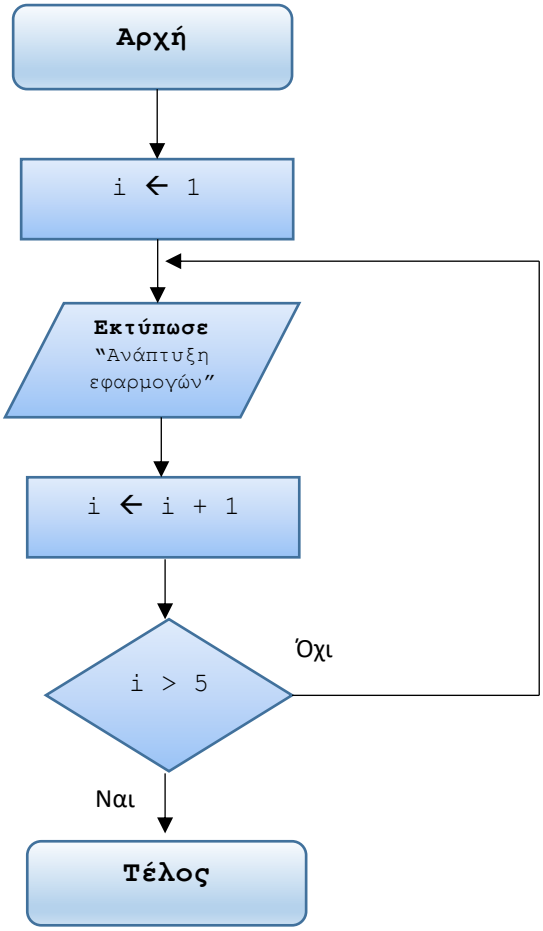
Προφανώς, η εντολή εκτύπωσης θα εκτελεστεί τουλάχιστον μία φορά, οπότε μπορούμε να χρησιμοποιήσουμε την `Αρχή_επανάληψης..Μέχρις_ότου`.

```
Αλγόριθμος Πολλαπλή_εκτύπωση_μηνύματος
    i ← 1
    Αρχή_επανάληψης
        Εκτύπωσε “Ανάπτυξη εφαρμογών”
        i ← i + 1
    Μέχρις_ότου i > 5
Τέλος Πολλαπλή_εκτύπωση_μηνύματος
```



Φυσικά, η λογική συνθήκη θα μπορούσε να είναι ισοδύναμα `i = 6` αντί `i > 5`.

Το αντίστοιχο λογικό διάγραμμα είναι το εξής:



Παράδειγμα 2: Να γραφτεί αλγόριθμος που διαβάζει μία ακολουθία αριθμών που αφορούν ημερήσιες εισπράξεις ενός καταστήματος. Αφού διαβαστούν όλοι οι αριθμοί (*εισπράξεις*) θα πρέπει να υπολογίσει το σύνολο των εισπράξεων του καταστήματος. Το τέλος της εισόδου των αριθμών θα σηματοδοτείται μόλις διαβαστεί ο αριθμός -1.

Ας δούμε τον αλγόριθμο σε ψευδογλώσσα, κάνοντας χρήση της `Αρχή_επανάληψης..Μέχρις_ότου`:

```
Αλγόριθμος Επαναληπτική_είσοδος_στοιχείων
    SUM ← 0
    Διάβασε Είσπραξη
    Αρχή_επανάληψης
        SUM ← SUM + Είσπραξη
        Διάβασε Είσπραξη
    Μέχρις_ότου Είσπραξη = -1
    Εκτύπωσε SUM
```

Η μεταβλητή SUM είναι ο συσσωρευτής. Πριν την επανάληψη πρέπει να πάρει αρχική τιμή 0.

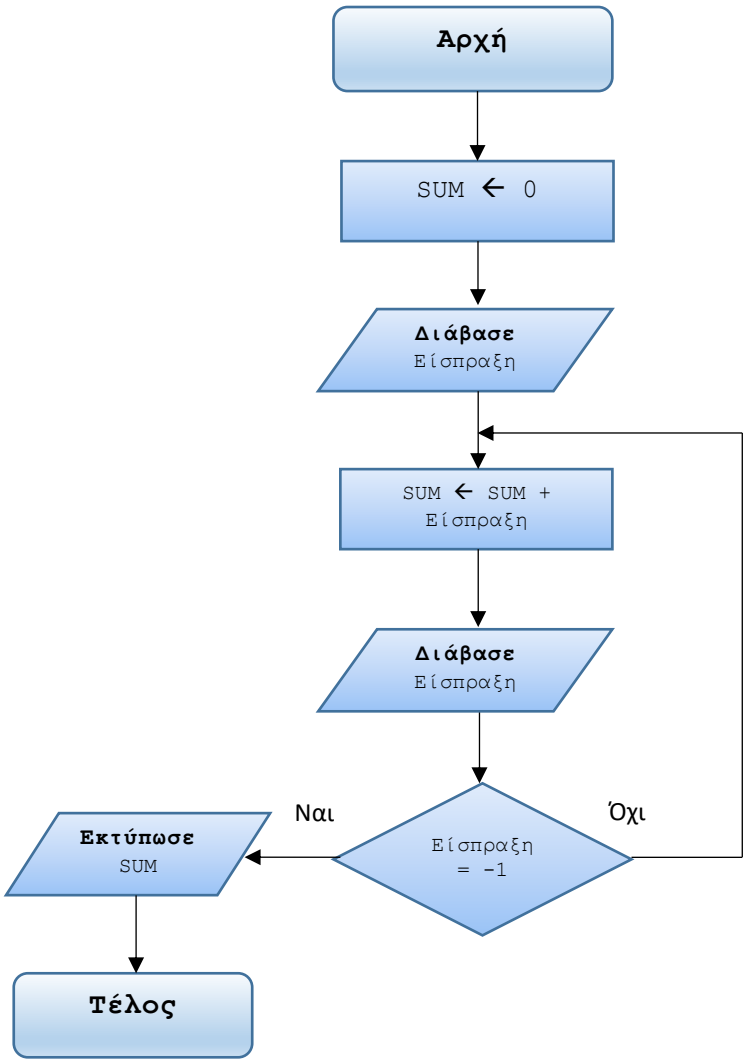
Τέλος Επαναληπτική_είσοδος_στοιχείων

Για να λειτουργήσει σωστά ο αλγόριθμος υπάρχει **μία απαραίτητη προϋπόθεση**: **Ο πρώτος αριθμός να μην είναι ο -1. Δηλαδή, υποθέτουμε ότι θα διαβαστεί μία τουλάχιστον είσπραξη**.

Επίσης, πριν την δομή επανάληψης Αρχή_επανάληψης...Μέχρις_ότου διαβάζεται ο πρώτος αριθμός, Αυτός θα αξιοποιηθεί μέσα στη δομή (εδώ προστίθεται σε έναν αθροιστή) και πριν το τέλος διαβάζεται ο επόμενος. Ακολούθως, γίνεται έλεγχος της συνθήκης κι εφόσον είναι ΨΕΥΔΗΣ πάει στο επόμενο loop. Αυτή είναι μία τυπική περίπτωση επαναληπτικής εισόδου δεδομένων με τη δομή Αρχή_επανάληψης..Μέχρις_ότου.

Υπάρχει περίπτωση να μην εκτελεστεί καθόλου η επανάληψη; Όχι, η φύση της δομής Αρχή_επανάληψης…Μέχρις_ότου είναι ότι οι εντολές θα εκτελεστούν τουλάχιστον μία φορά.

Το αντίστοιχο λογικό διάγραμμα είναι το εξής:



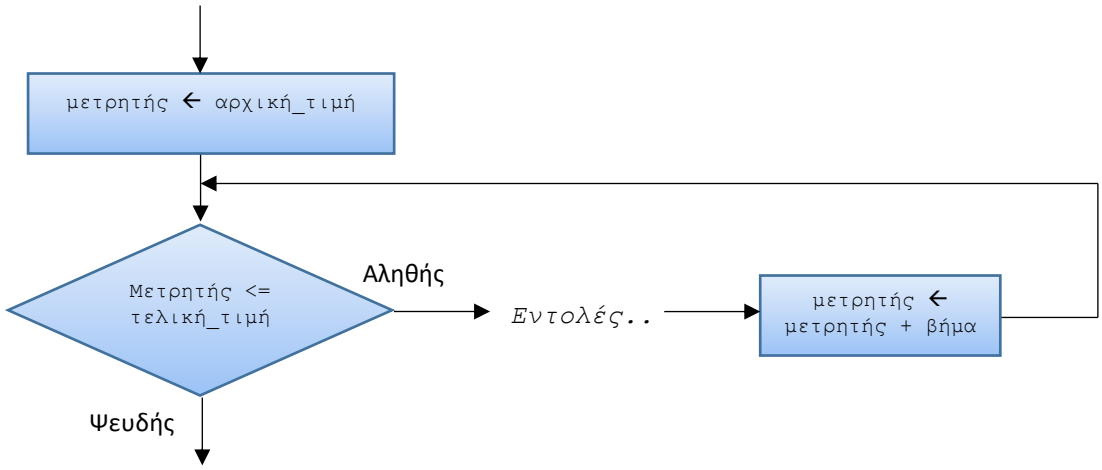
Διερευνήστε αν υπάρχει τρόπος να υλοποιηθεί ο παραπάνω αλγόριθμος της επαναληπτικής εισόδου δεδομένων χωρίς να χρησιμοποιήσουμε δύο φορές την εντολή **Διάβασε** Είσπραξη.

Για..από..μέχρι..με_βήμα

Πρόκειται για την **καταλληλότερη δομή επανάληψης όταν γνωρίζουμε τον αριθμό των επαναλήψεων.**

Μορφή:

```
Για μετρητής από αρχική_τιμή μέχρι τελική_τιμή με_βήμα βήμα
      Εντολές…
Τέλος_επανάληψης
```



Παρατηρήσεις:

- Χρειαζόμαστε οπωσδήποτε μία **μεταβλητή** για χρήση ως **μετρητής**. Συνήθως, του δίνουμε το όνομα `i` ή `j` κλπ.

- Ο μετρητής παίρνει μία αρχική τιμή. Στη συνέχεια, συγκρίνεται η τιμή του μετρητή με την τελική τιμή. Αν η τιμή του μετρητή δεν έχει ξεπεράσει την τελική τιμή τότε εκτελούνται οι εντολές. Η τιμή του μετρητή αυξάνεται κατά την τιμή του βήματος (συνήθως 1) και το loop συνεχίζεται.

- Στο διάγραμμα ροής, έχουμε θέσει ξεχωριστά *εντολή εκχώρησης αρχικής τιμής στον μετρητή καθώς και εντολή εκχώρησης αύξησης της τιμής του μετρητή κατά βήμα*. Στη δομή `Για..από..μέχρι..με_βήμα` δεν χρειάζεται γιατί **αποδίδονται αυτόματα**.

- Στην συνθήκη ελέγχουμε και για ισότητα, δηλαδή μικρότερο ή ίσο.

- Επειδή γίνεται πρώτα έλεγχος του μετρητή, αν έχει ξεπεράσει την τελική τιμή, **είναι δυνατόν οι εντολές να μην εκτελεστούν καθόλου**.

- **Το βήμα δεν μπορεί να είναι 0** διότι η επανάληψη δεν θα σταματήσει ποτέ, αφού ο μετρητής δεν θα αλλάζει τιμή.

- Προαιρετικά, **όταν το βήμα αύξησης είναι +1 τότε μπορούμε να παραλείψουμε το τμήμα `με_βήμα 1`, διότι αυτόματα υπονοείται**.

Ας ξαναδούμε το παράδειγμα με την εκτύπωση 5 φορές του μηνύματος “Ανάπτυξη εφαρμογών”.

Παράδειγμα 1: Να εκτυπωθεί 5 φορές το μήνυμα “Ανάπτυξη εφαρμογών”.

Επειδή ο αριθμός των επαναλήψεων είναι γνωστός (5) η δομή `Για..από..μέχρι..με_βήμα` είναι η καταλληλότερη:

```
Αλγόριθμος Πολλαπλή_εκτύπωση_μηνύματος
    Για i από 1 μέχρι 5 με_βήμα 1
        Εκτύπωσε “Ανάπτυξη εφαρμογών”
    Τέλος_επανάληψης
Τέλος Πολλαπλή_εκτύπωση_μηνύματος
```

Η μεταβλητή `i` είναι ο μετρητής. Ξεκινάει από το 1 και φτάνει μέχρι και το 5 με βήμα αύξησης 1.

Παράδειγμα 2: Να γραφτεί αλγόριθμος που διαβάζει μία ακολουθία αριθμών που αφορούν ημερήσιες εισπράξεις για τον μήνα Απρίλιο ενός καταστήματος. Αφού διαβαστούν όλοι οι αριθμοί (*εισπράξεις*) θα πρέπει να υπολογίσει το σύνολο των εισπράξεων του καταστήματος. Υποθέτουμε ότι το κατάστημα είναι ανοικτό κάθε ημέρα, για όλο τον μήνα.

Ο μήνας Απρίλιος έχει 30 ημέρες και εφόσον είναι ανοικτό όλες τις ημέρες έχουμε αντίστοιχα 30 ημερήσιες εισπράξεις. Συνεπώς, θα διαβαστούν επαναληπτικά 30 αριθμοί και κάθε φορά

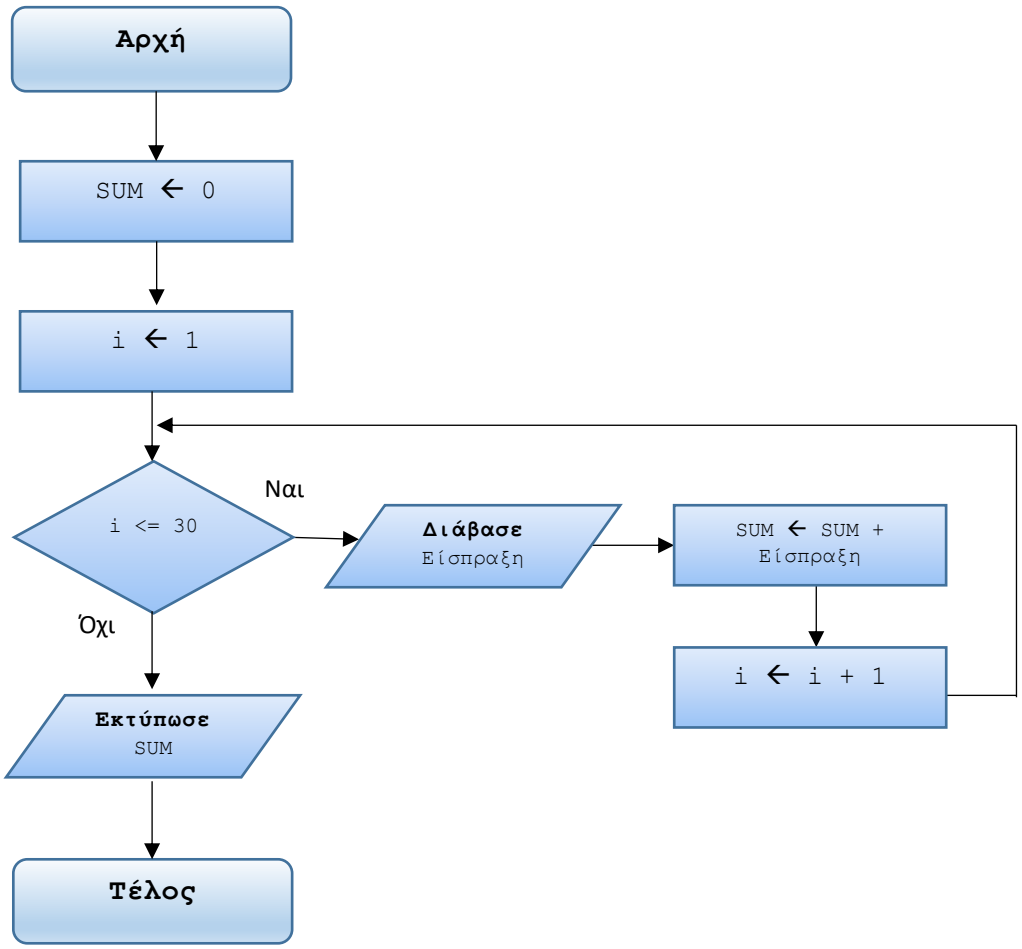
που διαβάζεται ένας αριθμός θα προστίθεται σε έναν συσσωρευτή (Sum). Στο τέλος των επαναλήψεων, ο συσσωρευτής θα μας έχει δώσει το άθροισμα των εισπράξεων του μήνα.

Ας δούμε τον αλγόριθμο σε ψευδογλώσσα:

```
Αλγόριθμος Επαναληπτική_είσοδος_στοιχείων
    SUM ← 0
    Για  i από 1 μέχρι 30 με_βήμα 1
        Διάβασε Είσπραξη
        SUM ← SUM + Είσπραξη
    Τέλος_επανάληψης
    Εκτύπωσε SUM
Τέλος Επαναληπτική_είσοδος_στοιχείων
```

Η μεταβλητή SUM είναι ο συσσωρευτής. Πριν την επανάληψη πρέπει να πάρει αρχική τιμή 0.

Το αντίστοιχο λογικό διάγραμμα είναι το εξής:



Παράδειγμα 3: Να τροποποιηθεί αλγόριθμος του Παραδείγματος 2 ώστε να τυπώνει και τον μέσο όρο (ΜΟ) των εισπράξεων του μήνα Απριλίου.

Σε κάθε περίπτωση, για να υπολογιστεί ο μέσος όρος (ΜΟ) πρέπει πρώτα να έχουμε βρει το άθροισμα, το οποίο στη συνέχεια διαιρείται με το πλήθος των στοιχείων (*εδώ εισπράξεων*).

```
Αλγόριθμος Επαναληπτική_είσοδος_στοιχείων
    SUM ← 0
    ΜΟ ← 0
    Για  i από 1 μέχρι 30 με_βήμα 1
        Διάβασε Είσπραξη
        SUM ← SUM + Είσπραξη
    Τέλος_επανάληψης
    ΜΟ ← SUM / 30
    Εκτύπωσε SUM, ΜΟ
Τέλος Επαναληπτική_είσοδος_στοιχείων
```

Αρχικές τιμές πριν την επανάληψη.

Μετά την επανάληψη, υπολογίζουμε τον ΜΟ.

Ο αναγνώστης μπορεί να τροποποιήσει ελαφρά το προηγούμενο διάγραμμα ροής, ώστε να συμπεριλάβει την εύρεση του ΜΟ.

*Ο υπολογισμός του **αθροίσματος** και **μέσου όρου** ενός πλήθους στοιχείων, αποτελεί μία από τις τυπικές και συνηθισμένες επεξεργασίες, ιδιαίτερα κατά τη χρήση δομών δεδομένων (π.χ. πίνακες), που θα δούμε στο Κεφάλαιο 3.*

Παράδειγμα 4: Να γραφτεί αλγόριθμος που τυπώνει όλους τους ζυγούς αριθμούς από το 100 μέχρι το 2.

Με άλλα λόγια, θέλουμε να τυπωθούν διαδοχικά οι αριθμοί 100, 98, 96, … , 6, 4, 2.

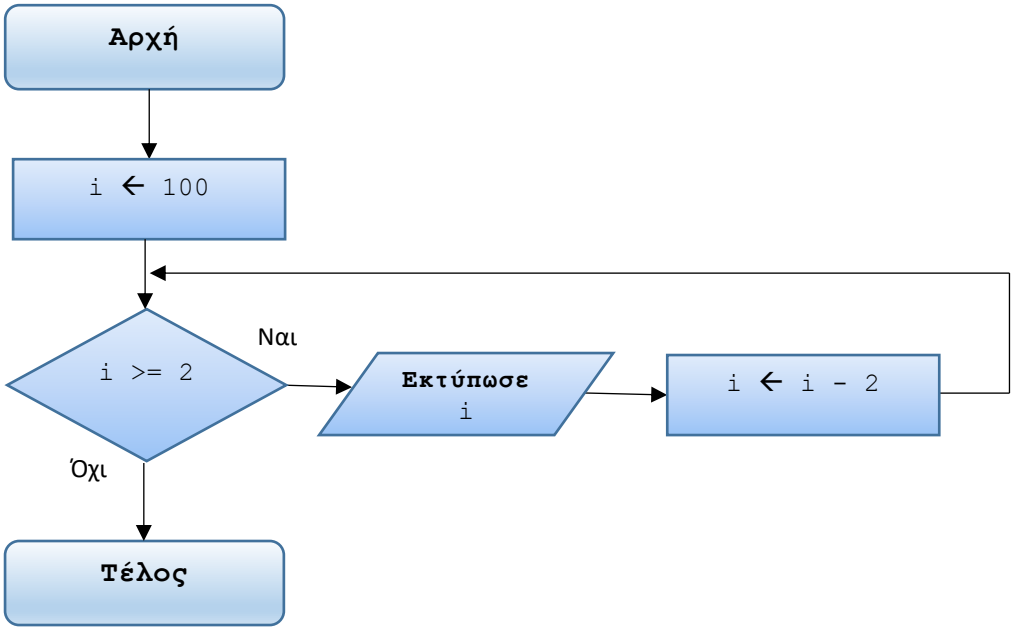
Πρόκειται για άλλη μία τυπική χρήση της επαναληπτικής δομής `Για..από..μέχρι..με_βήμα` αφού γνωρίζουμε την αρχική τιμή (`100`) την τελική τιμή (`2`) και το βήμα μεταβολής του μετρητή (`-2`).

```
Αλγόριθμος Εκτύπωση_ζυγών_κατά_φθίνουσα_σειρά
    Για  i από 100 μέχρι 2 με_βήμα -2
        Εκτύπωσε i
    Τέλος_επανάληψης
Τέλος Εκτύπωση_ζυγών_κατά_φθίνουσα_σειρά
```

Ουσιαστικά τυπώνουμε τις τιμές του μετρητή.

Φυσικά, θα μπορούσαν να χρησιμοποιηθούν ισοδύναμα και οι άλλες δύο δομές επανάληψης, κάτι που αφήνεται ως άσκηση στον αναγνώστη.

Το αντίστοιχο λογικό διάγραμμα φαίνεται παρακάτω:



Στο λογικό διάγραμμα, πρέπει να δηλώσουμε σαφώς την αρχική τιμή του μετρητή (i← 100), τη συνθήκη ελέγχου (i >= 2) και την μεταβολή του μετρητή κατά ένα ποσό (i ← i - 2).

Σύγκριση δομών επανάληψης

Στον παρακάτω πίνακα, παρουσιάζουμε εν συντομία τα χαρακτηριστικά των τριών αλγοριθμικών δομών επανάληψης.

Όσο..επανάλαβε	Αρχή_επανάληψης..Μέχρις_ότου	Για..από..μέχρι..με_βήμα
Κατάλληλη για χρήση όταν δεν γνωρίζουμε τον αριθμό των επαναλήψεων (*π.χ. επαναληπτική είσοδος στοιχείων*).	Κατάλληλη για χρήση όταν δεν γνωρίζουμε τον αριθμό των επαναλήψεων (*π.χ. επαναληπτική είσοδος στοιχείων*).	Κατάλληλη για χρήση όταν γνωρίζουμε τον αριθμό των επαναλήψεων.
Οι εντολές εκτελούνται 0, 1 ή περισσότερες φορές.	Οι εντολές εκτελούνται 1 ή περισσότερες φορές.	Οι εντολές εκτελούνται 0, 1 ή περισσότερες φορές.
Οι εντολές εκτελούνται συνεχώς όσο η συνθήκη είναι ΑΛΗΘΗΣ	Οι εντολές εκτελούνται συνεχώς όσο η συνθήκη είναι ΨΕΥΔΗΣ	Οι εντολές εκτελούνται όσο η τιμή του μετρητή δεν έχει ξεπεράσει την τελική τιμή (μέχρι).
Ο μετρητής δεν είναι απαραίτητος *(εξαρτάται από το πρόβλημα).*	Ο μετρητής δεν είναι απαραίτητος *(εξαρτάται από το πρόβλημα).*	Ο μετρητής είναι απαραίτητος.

Ο μετρητής μπορεί λάβει και πραγματικές τιμές. Για παράδειγμα:

```
Για i από 0.5 μέχρι 0.9 με_βήμα 0.1
```

Πολλαπλασιασμός αλά ρωσικά

Είναι **μία μέθοδος πολλαπλασιασμού δύο αριθμών που χρησιμοποιείται ιδιαίτερα στους υπολογιστές**[1]. Συγκεκριμένα, αν έχουμε δύο αριθμούς X και Y και θέλουμε το X * Y τότε αυτό γίνεται ως εξής:

Βήμα 1: Ο X διπλασιάζεται και ο Y υποδιπλασιάζεται.

Βήμα 2: Συνεχίζουμε το Βήμα 1 μέχρι να φτάσει ο Y σε 1.

Βήμα 3: Το γινόμενο θα είναι το άθροισμα των στοιχείων X στα βήματα που το Y είναι περιττός.

Π.χ. **X** = 45, **Y** = 19

X	**Y**	
45	19	45
90	9	90
180	4	
360	2	
720	1	720
	Άθροισμα =	**855**

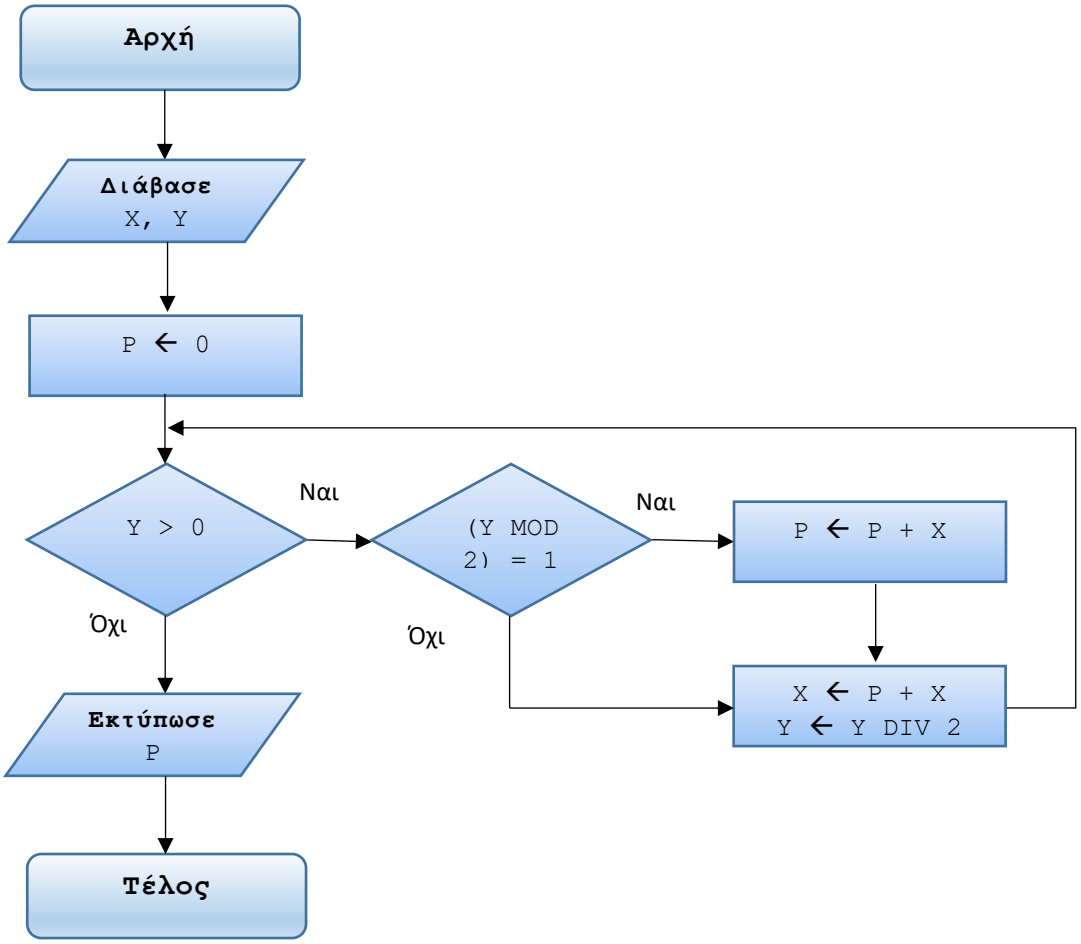
Σε ψευδογλώσσα, θα γράψουμε ως εξής τον αλγόριθμο:

```
Αλγόριθμος Πολλαπλασιασμός_αλά_ρωσικά
    Διάβασε X, Y     !Τα δεδομένα που εισάγουμε. Σχόλιο.
    P ← 0            !Αρχική τιμή γινομένου (αρχικοποίηση μεταβλητής)
    Όσο Y > 0 επανάλαβε
        Αν (Y MOD 2) = 1 τότε   !αν το υπόλοιπο της διαίρεσης του Y με το 2 είναι 1
            P ← P + X           !πρόσθεσε το X στο άθροισμα
        Τέλος_αν
        X ← X * 2               !διπλασιασμός του X
        Y ← Y DIV 2             !υποδιπλασιασμός του Y. Παίρνουμε το πηλίκο του Y με το 2
```

[1] Δες παράρτημα.

```
        Τέλος_επανάληψης
        Εκτύπωσε P      !Αποτέλεσμα
Τέλος Πολλαπλασιασμός_αλά_ρωσικά
```

Το αντίστοιχο διάγραμμα ροής είναι:



Ερωτήσεις κατανόησης

1. Τί εννοούμε με τον όρο αλγόριθμο;

2. Αναφέρατε τα κριτήρια ενός σωστού αλγορίθμου.

3. Η επιστήμη της Πληροφορικής μελετά τους αλγορίθμους από διάφορες οπτικές γωνίες. Εξηγήστε κάθε μία.

4. Με ποιούς τρόπους γίνεται η αναπαράσταση (περιγραφή) ενός αλγορίθμου. Από αυτούς, ποιός είναι ο πιο δομημένος τρόπος αναπαράστασης;

5. Οι μεταβλητές είναι βασικό στοιχείο των αλγορίθμων στην περιγραφή κυρίως με ψευδογλώσσα και λογικό διάγραμμα.
   a. Πρέπει τα ονόματά τους να είναι μοναδικά σε όλον τον αλγόριθμο;
   b. Πρέπει οι τιμές τους ή το όνομά τους να είναι σταθερό κατά τη διάρκεια εκτέλεσης του αλγορίθμου;
   c. Πώς ονομάζεται η εντολή που τοποθετεί μία τιμή σε μία μεταβλητή;
   d. Μπορεί μία μεταβλητή να έχει ταυτόχρονα δύο διαφορετικές τιμές την ίδια στιγμή κατά τη διάρκεια εκτέλεσης του αλγορίθμου;
   e. Μία αριθμητική μεταβλητή, μπορεί να διακρίνεται περαιτέρω σε …….. ή …… , ανάλογα με το είδος (τύπο) του αριθμητικού δεδομένου που αποθηκεύει.

6. Αν `X ← 5` και `Y ← 2` τί αποτίμηση θα έχουν οι παρακάτω λογικές εκφράσεις:

   a. `X >= 5 ΚΑΙ X < 5`

   b. `X >= 5 Ή X < 5`

   c. `ΟΧΙ (X = 5)`

   d. `X = 5 ΚΑΙ Y = 2`

   e. `ΟΧΙ (X = 5) Ή Y > 2`

   f. `((X < 5) Ή (Y = 2)) Ή (X = 5)`

   g. `ΟΧΙ (X <> 5) ΚΑΙ ΟΧΙ (Y <> 2)`

7. Ποιές είναι οι βασικές εντολές της ψευδογλώσσας ;

8. Πότε χρησιμοποιούμε:

a. Την ακολουθιακή δομή.
b. Την δομή επιλογής **Αν..τότε**
c. Την δομή επιλογής **Αν..τότε..αλλιώς**

9. Υπάρχει περίπτωση στη δομή επιλογής **Αν..τότε..αλλιώς** να εκτελεστούν ταυτόχρονα οι εντολές μετά το **τότε** και το **αλλιώς**;

10. Πότε είναι κατάλληλη η χρήση της δομή πολλαπλής επιλογής;

11. Πώς ονομάζεται η περίπτωση που μία δομή επιλογής βρίσκεται εντός μίας άλλης δομής επιλογής;

12. Υπάρχουν 3 δομές επανάληψης. Μπορείτε να αναφέρετε σε ποιές περιπτώσεις είναι καταλληλότερη η καθεμία;

13. Ένας φίλος σας, σας αναφέρει ότι θέλει να επιλύσει ένα πρόβλημα που αφορά επεξεργασίες 100 αριθμών. Ποιά δομή επανάληψης θα προτιμήσετε;

14. Πάλι ο φίλος σας ισχυρίζεται ότι στη δομή **Αρχή_επανάληψης..Μέχρις_ότου** οι εντολές θα εκτελεστούν οπωσδήποτε μία φορά. Έχει δίκιο;

15. Ο φίλος σας έχει εκνευριστεί γιατί δεν μπορεί να βρει το λάθος στον παρακάτω αλγόριθμο:

```
Αλγόριθμος Εκτύπωση_100_αριθμών
      i ← 1
      Όσο i <= 100 επανάλαβε
            Εκτύπωσε i
      Τέλος_επανάληψης
Τέλος Εκτύπωση_100_αριθμών
```

Πράγματι, κάτι δεν πάει καλά. Θα τον βοηθήσετε;

16. Σε συνέχεια του προηγούμενου, αφού τον βοηθήσετε, κατασκευάστε και το αντίστοιχο λογικό διάγραμμα.

17. Πολλές φορές χρειάζεται κατά την είσοδο ενός δεδομένου να ελέγξουμε αν αυτό είναι έγκυρο. Για παράδειγμα, κατά την είσοδο ενός βαθμού μαθητή πρέπει να ελέγξουμε αν είναι μεταξύ 1 και 20. Κι επίσης, πρέπει ο έλεγχος να γίνεται συνεχώς μέχρι ο βαθμός να είναι έγκυρος (*μπορεί να εισάγεται επανειλημμένα ένας μη*

έγκυρος βαθμός, οπότε πρέπει να προβλέπεται ένας τέτοιος συνεχής έλεγχος). Γράψτε ένα κομμάτι αλγορίθμου που εκτελεί αυτόν τον έλεγχο εγκυρότητας του βαθμού (*υπόδειξη : θα είναι μία δομή επανάληψης*).

18. Στο παρακάτω κομμάτι αλγορίθμου, ο μετρητής `i` ξεκινάει από το `1` και φτάνει μέχρι το `20` με βήμα `3`. Πόσες φορές θα εκτελεστεί η εντολή στη δομή επανάληψης;

```
Για i από 1 μέχρι 20 με_βήμα 3
      Εκτύπωσε i
Τέλος_επανάληψης
```

19. Ποιά η ιδιαίτερη αξία του πολλαπλασιασμού αλά ρωσικά για τους υπολογιστές; (*πρέπει να έχετε διαβάσει το παράρτημα*).

20. Στο παράρτημα, δίνονται τα βήματα για τη μετατροπή ενός αριθμού από το δεκαδικό σύστημα στο δυαδικό. Μπορείτε να γράψετε τα βήματα σε αλγόριθμο σε ψευδογλώσσα και διάγραμμα ροής.

Παράρτημα

Ποιά η ιδιαίτερη αξία του πολλαπλασιασμού αλά ρωσικά για τους υπολογιστές

Ο πολλαπλασιασμός αλά ρωσικά δύο αριθμών περιλαμβάνει μία συνεχή διαδικασία όπου ο πρώτος διπλασιάζεται και ο δεύτερος υποδιπλασιάζεται.

Ο διπλασιασμός και υπο-διπλασιασμός ενός αριθμού πραγματοποιείται στα κυκλώματα του ΗΥ με έναν συγκεκριμένο τρόπο: αυτόν της **ολίσθησης** (Shift).

Ως γνωστόν, τα δεδομένα στο εσωτερικό του ΗΥ αποθηκεύονται ως μία σειρά από δυαδικά ψηφία (0 και 1). Κι αυτό, ανεξάρτητα το πώς έχουν οριστεί από τον προγραμματιστή π.χ. ακέραιος, πραγματικός κ.λπ. Για παράδειγμα, ο αριθμός `15` στο δεκαδικό σύστημα αρίθμησης κωδικοποιείται στο δυαδικό ως `1111` κι αν χρησιμοποιήσουμε 8 bits (1 byte) τότε αποθηκεύεται ως `00001111`.

Τώρα, ο διπλασιασμός του `15` είναι το `30`, στο δυαδικό ο **διπλασιασμός του `00001111` πραγματοποιείται αν μετακινήσουμε(*ολισθήσουμε*) τα ψηφία κατά μία θέση αριστερά, προσθέτοντας ένα 0 στο τέλος και αγνοώντας το πρώτο** `0`. Έτσι, προκύπτει ο `00011110`.

Διπλασιασμός του 15:

Δεκαδικό σύστημα : `15 x 2 = 30`

Δυαδικό σύστημα: `00001111` ολίσθηση αριστερά = `000011110` → `00011110`

Επίσης, ο υποδιπλασιασμός (ακέραια διαίρεση) του `15` είναι το `7`, στο δυαδικό ο **υποδιπλασιασμός του `00001111` πραγματοποιείται αν μετακινήσουμε(*ολισθήσουμε*) τα ψηφία κατά μία θέση δεξιά, προσθέτοντας ένα 0 στην αρχή και αγνοώντας το τελευταίο** `1`. Έτσι, προκύπτει ο `0000111`.

Υποδιπλασιασμός του 15 (ακέραια διαίρεση):

Δεκαδικό σύστημα : `15 \ 2 = 7`

Δυαδικό σύστημα: `00001111` ολίσθηση δεξιά = `000001111` → `00000111`

Μετατροπή δεκαδικού αριθμού σε δυαδικό

Για τη μετατροπή ενός αριθμού από το δεκαδικό σύστημα αρίθμησης στο δυαδικό εκτελούμε την εξής διαδικασία:

Βήμα 1: Διαιρούμε τον αριθμό με το 2.
Βήμα 2: Κρατάμε το υπόλοιπο (θα είναι 1 ή 0).
Βήμα 3: Παίρνουμε το πηλίκο και επαναλαμβάνουμε από το Βήμα 1. Αν το πηλίκο είναι 0 πηγαίνουμε στο Βήμα 4.
Βήμα 4: Βάζουμε στη σειρά το ένα μετά το άλλο τα υπόλοιπα που βρήκαμε, ξεκινώντας από το τελευταίο που βρήκαμε προς το πρώτο (δηλαδή, κατ' αντίστροφη σειρά εύρεσης).

Παράδειγμα 1: Ο αριθμός 15

Διαίρεση	Πηλίκο	Υπόλοιπο
15 / 2	7	1
7 / 2	3	1
3 / 2	1	1
1 / 2	0	1

Ο δυαδικός αντίστοιχος του 15 είναι ο 1111.

Παράδειγμα 2: Ο αριθμός 14

Διαίρεση	Πηλίκο	Υπόλοιπο
14 / 2	7	0
7 / 2	3	1
3 / 2	1	1
1 / 2	0	1

Ο δυαδικός αντίστοιχος του 14 είναι ο 1110.

- Δοκιμάστε μόνοι σας για τους αριθμούς 25 και 37.

Μετατροπή δυαδικού αριθμού σε δεκαδικό

Για τη μετατροπή ενός αριθμού από το δυαδικό σύστημα αρίθμησης στο δεκαδικό εκτελούμε την εξής διαδικασία:

Βήμα 1: Πολλαπλασιάζουμε κάθε ψηφίο του δυαδικού αριθμού (0 ή 1) με το 2 εις τη δύναμη της θέσης του ψηφίου (δηλαδή, αν το ψηφίο είναι στην πρώτη θέση από δεξιά τότε είναι 2^0, αν είναι στην δεύτερη θέση τότε είναι 2^1 κ.ο.κ.)
Βήμα 2: Αθροίζουμε τα γινόμενα όλων αυτών των πολλαπλασιασμών.

Τα παραπάνω είναι, σε γενικές γραμμές, η εφαρμογή του εξής τύπου:

$$abc = a \times 2^2 + b \times 2^1 + c \times 2^0$$

Παράδειγμα 1: Ο αριθμός 1111

$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 4 + 2 + 1 = 15$

Ο δεκαδικός αντίστοιχος του 1111 είναι ο 15.

Παράδειγμα 2: Ο αριθμός 1110

$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 4 + 2 + 0 = 14$

Ο δεκαδικός αντίστοιχος του 1110 είναι ο 14.

- Δοκιμάστε μόνοι σας για τους αριθμούς 11001 και 100101.

ΤΕΛΟΣ ΣΗΜΕΙΩΣΕΩΝ ΚΕΦΑΛΑΙΟΥ 2