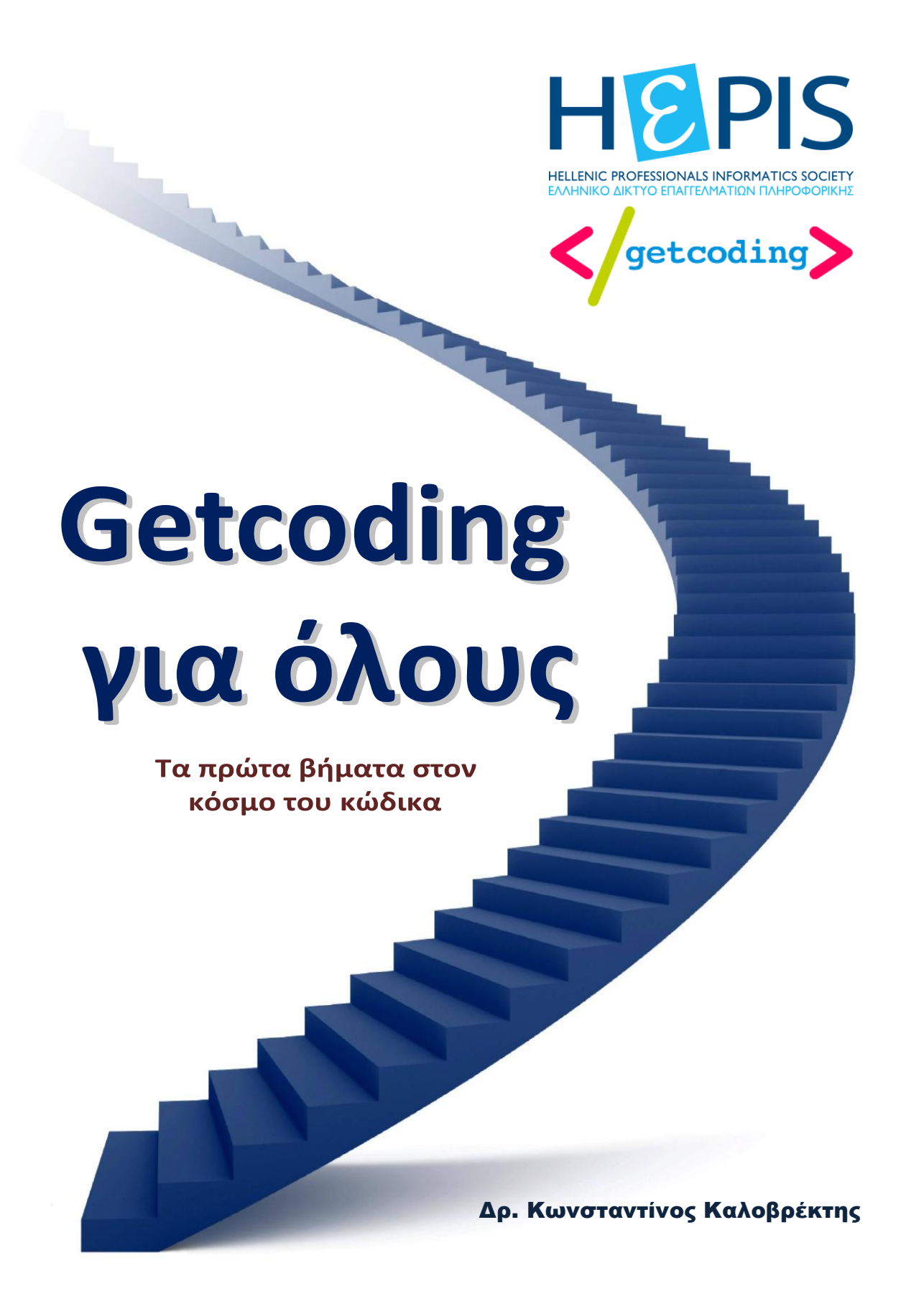

HEPIS
HELLENIC PROFESSIONALS INFORMATICS SOCIETY
ΕΛΛΗΝΙΚΟ ΔΙΚΤΥΟ ΕΠΑΓΓΕΛΜΑΤΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
</getcoding>
Getcoding
για όλους
Τα πρώτα βήματα στον
κόσμο του κώδικα
Δρ. Κωνσταντίνος Καλοβρέκτης

Getcoding για όλους

Τα πρώτα βήματα στον κόσμο του κώδικα

Δρ. Κωνσταντίνος Καλοβρέκτης

Σύμβουλος & Επιστημονικός Υπεύθυνος Εκπαίδευσης του Getcoding της HePIS

Τίτλος πρωτοτύπου:

Getcoding για όλους
Τα πρώτα βήματα στον κόσμο του κώδικα

ISBN 978-960-93-6991-6

Εξώφυλλο (εικόνα) The Next Level Business Solutions
Οπισθόφυλλο(κείμενο) http://www.getcoding.gr

Αγαπητέ αναγνώστη,

Κάθε αλληλεπίδραση ανάμεσα σε άνθρωπο και υπολογιστή γίνεται μέσα από τον κώδικα. Είτε χρησιμοποιείς τον υπολογιστή σου, την ταμπλέτα, το κινητό, ή οποιαδήποτε άλλη ηλεκτρονική συσκευή ο κώδικας είναι εκεί! Μπορεί να μην τον βλέπεις, αλλά ο κώδικας είναι εκεί. Αποτελεί αναπόσπαστο κομμάτι της καθημερινότητας, βρίσκεται παντού και είναι απαραίτητος για την καθημερινή επαφή με την τεχνολογία.

Στοιχειώδεις δεξιότητες προγραμματισμού θα απαιτούνται πλέον σε κάθε τομέα του παραγωγικού ιστού. Τομείς, όπως αυτοί της Τραπεζικής, της Ιατρικής ή της Δημοσιογραφίας, θα έχουν ως απαραίτητη προϋπόθεση βασικές γνώσεις κώδικα και προγραμματισμού. Ο Προγραμματισμός σε λίγα χρόνια θα αποτελεί απαραίτητη δεξιότητα όπως αυτή της γραφής και της ανάγνωσης.

Η απόκτηση βασικών δεξιοτήτων Προγραμματισμού αναπτύσσει δεξιότητες όπως η επίλυση προβλημάτων, η κριτική σκέψη, η λογική, η δημιουργικότητα καθώς και η αλγοριθμική σκέψη.

Στόχος του βιβλίου αλλά και της διαδικτυακής εφαρμογής είναι να φέρει τους συμμετέχοντες πιο κοντά στον Προγραμματισμό μέσα από έναν εύκολο και διασκεδαστικό τρόπο. Αποτελεί ένα χρήσιμο εργαλείο στα χέρια των γονέων και των εκπαιδευτικών. Είναι άρρηκτα συνδεδεμένο με τη διαδικτυακή εφαρμογή του Getcoding.gr μέσα από την οποία οι συμμετέχοντες αποκτούν βασικές έννοιες Προγραμματισμού όπως οι επαναλήψεις, οι υποθέσεις, οι ερωτήσεις και τα διαγράμματα ροής. Η πρωτοβουλία Getcoding.gr υλοποιείται από το Ελληνικό Δίκτυο Επαγγελματιών Πληροφορικής (HePIS) ο οποίος αποτελεί τον φορέα εκπροσώπησης των Ελλήνων Επαγγελματιών Πληροφορικής και έχει ως αποστολή τη προώθηση και διάχυση της Πληροφορικής στην ελληνική κοινωνία. Επιπρόσθετα, η HePIS είναι μέλος του Ευρωπαϊκού Συμβουλίου Επαγγελματικών Ενώσεων Πληροφορικής (CEPIS).

Εύχομαι το παρόν βιβλίο να αποτελέσει έναν σημαντικό σύμμαχο στα πρώτα σου βήματα στον μαγικό κόσμο του Προγραμματισμού.

Καλή ανάγνωση,

Νίκος Φαλδαμής
Πρόεδρος HePIS

Σχετικά με τον Συγγραφέα

Ο **Κωνσταντίνος Καλοβρέκτης** είναι Σύμβουλος και Επιστημονικός Υπεύθυνος Εκπαίδευσης του Getcoding της HePIS. Είναι Μεταδιδακτορικός ερευνητής (PostDoc) και κάτοχος Διδακτορικού στην Πληροφορική (Ph.D). Κατέχει μεταπτυχιακή ειδίκευση σε Προηγμένα Πληροφοριακά Συστήματα (M.Sc) και μεταπτυχιακή ειδίκευση στις Επιστήμες της Αγωγής (M.A.Ed), ενώ επίσης έχει βραβευτεί από την Microsoft για την ανάπτυξη καινοτόμων εκπαιδευτικών δραστηριοτήτων στο πεδίο των ΤΠΕ.

Στο βιογραφικό του έχει τη συγγραφή πλήθους ακαδημαϊκών συγγραμμάτων καθώς και σημαντικό αριθμό εργασιών σε έγκριτα διεθνή επιστημονικά περιοδικά και συνέδρια, με πλήρη κρίση. Επίσης, έχει πολυετή εμπειρία σε ερευνητικά προγράμματα με αντικείμενο την ανάπτυξη κώδικα σε συστήματα μετρήσεων & ελέγχου, καθώς και έργο στην ανάπτυξη εκπαιδευτικών εφαρμογών μεθοδολογίας κατά STEM (Science, Technology, Engineering and Mathematics).

Τέλος, είναι κριτής σε διεθνή επιστημονικά περιοδικά και συνέδρια, συντάκτης σε διεθνή περιοδικό, ενώ διαθέτει επαγγελματική εμπειρία σε διαφορετικά περιβάλλοντα εκπαίδευσης, αλλά και σε οργανισμούς και επιχειρήσεις.

e-mail: kkalovr@uth.gr

Υπεύθυνος επικοινωνίας της HePIS

Μανώλης Λαμπόβας
Τηλ. 210-3729161
e-mail: Manolis.Labovas@hepis.gr

Οι ήρωες του Getcoding

Ίρις

Θαλής

ΠΕΡΙΕΧΟΜΕΝΑ

1

Η επιστήμη της πληροφορικής

1.1 Εισαγωγή

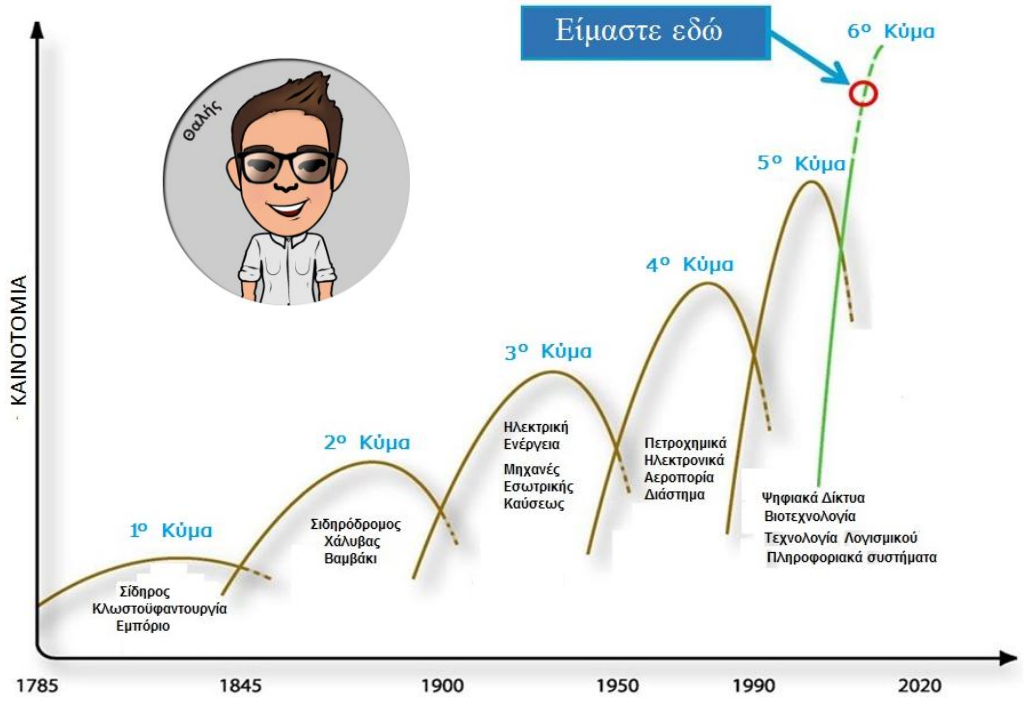
Η ραγδαία ανάπτυξη της τεχνολογίας τα τελευταία χρόνια δημιούργησε την ανάγκη της χρήσης υπολογιστών σε πολλούς τομείς της καθημερινότητάς μας. Είναι πλέον γεγονός πως σχεδόν σε κάθε δραστηριότητα μικρών και μεγάλων, από την απλή πληροφόρηση και ενημέρωση μέχρι τη διενέργεια σοβαρών και υπεύθυνων διαδικασιών, υπάρχει ένας υπολογιστής που εκτελεί ένα σύνολο εργασιών.

Η ιστορία των υπολογιστών όμως δεν ξεκινά με την έναρξη της ραγδαίας τεχνολογικής ανάπτυξης των τελευταίων χρόνων, αλλά από το μακρινό παρελθόν, καθώς οι πρόγονοί μας προσπάθησαν να δημιουργήσουν διάφορες μηχανές ώστε να διευκολύνουν καθημερινούς υπολογισμούς με εφαρμογές κυρίως στο εμπόριο και την αστρονομία.

Ένα πολύ σημαντικό παράδειγμα είναι και ο περίφημος μηχανισμός των Αντικυθήρων (Σχήμα 1.1), που πολύ πιθανόν χρησιμοποιήθηκε ως υπολογιστής προσδιορισμού της θέσης των πλανητών για τον προσανατολισμό των πλοίων.

Οι πρόγονοί μας έθεσαν τις βάσεις, κυρίως θεωρητικές, για την κατασκευή των σημερινών υπολογιστών. Ανέπτυξαν θεωρίες και έκαναν μελέτες στις οποίες βασίστηκε η σημερινή επιστήμη των υπολογιστών. Έτσι θα μπορούσαμε να πούμε πως η εφαρμοσμένη πληροφορική βασίστηκε στο θεωρητικό υπόβαθρο που έθεσαν οι πρόγονοί μας, τη θεωρητική πληροφορική. Στο γράφημα του Σχήματος 1.2 απεικονίζονται τα κύματα τεχνολογίας στο χρόνο.

Σχήμα 1.1: Μηχανισμός Αντικυθήρων.



Σχήμα 1.2: Κύματα εξέλιξης των καινοτομιών τεχνολογίας.

Η επιστήμη των υπολογιστών ασχολείται με τη μελέτη και την ανάλυση θεωριών για τη δημιουργία μεθόδων (θεωρητική πληροφορική) που θα χρησιμοποιηθούν για ανάπτυξη εφαρμογών σε διάφορους τομείς των θετικών και τεχνολογικών επιστημών (εφαρμοσμένη πληροφορική).

1.2 Θεωρητική επιστήμη των υπολογιστών

Η θεωρητική επιστήμη των υπολογιστών ασχολείται με τη μελέτη και την ανάλυση θεωρητικών εννοιών της επιστήμης των υπολογιστών. Η έννοια και η φύση των πληροφοριών, η δομή των προγραμμάτων, η βελτιστοποίησή τους, η απόδοσή τους, η μαθηματική λογική, η κρυπτογραφία, η υπολογιστική γεωμετρία και πολλές άλλες θεωρητικές έννοιες αποτελούν το αντικείμενο της θεωρητικής επιστήμης των υπολογιστών.

Είναι μια αρκετά δύσκολη επιστήμη, που απαιτεί καλό γνωστικό υπόβαθρο σε διάφορους τομείς των θετικών επιστημών. Από αυτή την επιστήμη πηγάζουν όλες οι θεωρίες και οι τεχνικές που χρησιμοποιούνται σήμερα στους πραγματικούς υπολογιστές. Για παράδειγμα, ο τρόπος που θα εκτελεστεί ένα πρόγραμμα σε έναν υπολογιστή είναι προϊόν της επιστήμης της θεωρητικής πληροφορικής. Μέσα από θεωρητικούς υπολογισμούς και ειδικές μελέτες που βασίζονται σε μαθηματικά μοντέλα και θεωρίες, αναπτύσσονται μέθοδοι που χρησιμοποιούνται σε πραγματικές εφαρμογές υπολογιστών, δημιουργώντας έτσι την εφαρμοσμένη επιστήμη των υπολογιστών.

1.3 Εφαρμοσμένη επιστήμη των υπολογιστών

Η εφαρμοσμένη επιστήμη των υπολογιστών ασχολείται με ανάπτυξη υπολογιστικών εφαρμογών σε διάφορους επιστημονικούς τομείς, όπως της οικονομίας, της διοίκησης, της εκπαίδευσης, της βιομηχανίας κ.ά. Γενικότερα, η εφαρμοσμένη επιστήμη των υπολογιστών στοχεύει στην εύρεση εννοιών της επιστήμης των υπολογιστών που θα χρησιμοποιηθούν για την επίλυση ενός συνόλου προβλημάτων του πραγματικού κόσμου. Παραδείγματα εφαρμογών είναι τα προγράμματα γραφείου (Word, Excel, PowerPoint, Acrobat Reader κ.ά.), που χρησιμοποιούνται κυρίως στον εργασιακό τομέα και μας επιτρέπουν να συντάσσουμε κείμενα, να διαχειριζόμαστε

λογιστικά φύλλα, να δημιουργούμε παρουσιάσεις και να διαβάζουμε ηλεκτρονικά βιβλία (e-books) με τη χρήση ηλεκτρονικού υπολογιστή. Εφαρμογές της επιστήμης των υπολογιστών δεν θεωρούνται μόνο τα προγράμματα (λογισμικά), αλλά και οποιαδήποτε άλλα συστήματα που συμπεριλαμβάνουν στη δομή τους έναν ή περισσότερους ηλεκτρονικούς υπολογιστές.

Θα μπορούσαμε να πούμε πως η κατηγοριοποίηση των υπολογιστικών συστημάτων είναι ασταθής, ειδικότερα μεταξύ των υπολογιστικών συστημάτων γενικού σκοπού και των ενσωματωμένων υπολογιστικών συστημάτων και αυτό γιατί τα ενσωματωμένα υπολογιστικά συστήματα τείνουν να πλησιάσουν τις επιδόσεις των υπολογιστικών συστημάτων γενικού σκοπού. Ένα απλό παράδειγμα είναι τα έξυπνα τηλέφωνα (smartphones), της Apple, όπου οι επιδόσεις τους σε επεξεργαστική ισχύ και σε δυνατότητες εκτέλεσης λογισμικού έχουν πλησιάσει τόσο κοντά τις επιδόσεις ενός προσωπικού υπολογιστή (PC) που θα μπορούσε κανείς να εντάξει αυτή τη συσκευή στην κατηγορία των υπολογιστικών συστημάτων γενικού σκοπού.

Επειδή όμως έχει σχεδιαστεί κυρίως για χρήση ως τηλεφωνική συσκευή από το ευρύ κοινό, κατηγοριοποιείται, από τη συνιστώσα του πεδίου εφαρμογής της, στα ενσωματωμένα υπολογιστικά συστήματα. Τα υπολογιστικά συστήματα (υλικό) των ερευνητικών και επιστημονικών εφαρμογών προς το παρόν θα είναι οι μόνοι διεκδικητές της κατηγορίας για αρκετά χρόνια ακόμα για το λόγο ότι η τεχνολογία που χρησιμοποιείται για να επιτευχθούν οι επιθυμητές επιδόσεις κοστίζει υπερβολικά. Επιπλέον, οι επιδόσεις που παρέχουν τα συστήματα αυτής της κατηγορίας ξεπερνούν κατά πολύ τις απαιτήσεις των περισσότερων χρηστών.

Σχήμα 1.3: Υπολογιστικά συστήματα.

Κάνοντας μία έρευνα ανάμεσα στους βασικότερους τομείς των επιστημών που χρησιμοποιούν υπολογιστικά συστήματα θα μπορούσαμε να πούμε πως υπάρχουν οι τρεις παρακάτω κατηγορίες:

- Ενσωματωμένα υπολογιστικά συστήματα
- Υπολογιστικά συστήματα γενικού σκοπού
- Υπολογιστικά συστήματα επιστημονικών & ερευνητικών εφαρμογών

Στον πίνακα 1.1 συνοψίζουμε τις τρεις κατηγορίες υπολογιστικών συστημάτων αποδίδοντας το βαθμό του κόστους και της επίδοσης τους καθώς και τα πεδία εφαρμογών που αυτές ανήκουν.

Πίνακας 1.1: Κατηγορίες υπολογιστικών συστημάτων.

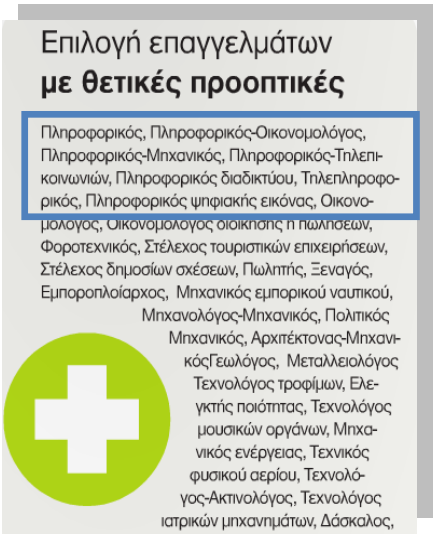
Κατηγορία	Επίδοση	Κόστος	Πεδίο εφαρμογών
Ενσωματωμένα υπολογιστικά συστήματα	3	3	Κινητά τηλέφωνα, Κονσόλες VideoGame, MP3.
Υπολογιστικά συστήματα γενικού σκοπού	4	4	Personal Computers (PC), Laptops, NetBooks, Tablet PC's κ.τ.λ..
Υπολογιστικά συστήματα επιστημονικών και ερευνητικών εφαρμογών	8	8	MainFrames, Super Computers κ.τ.λ..

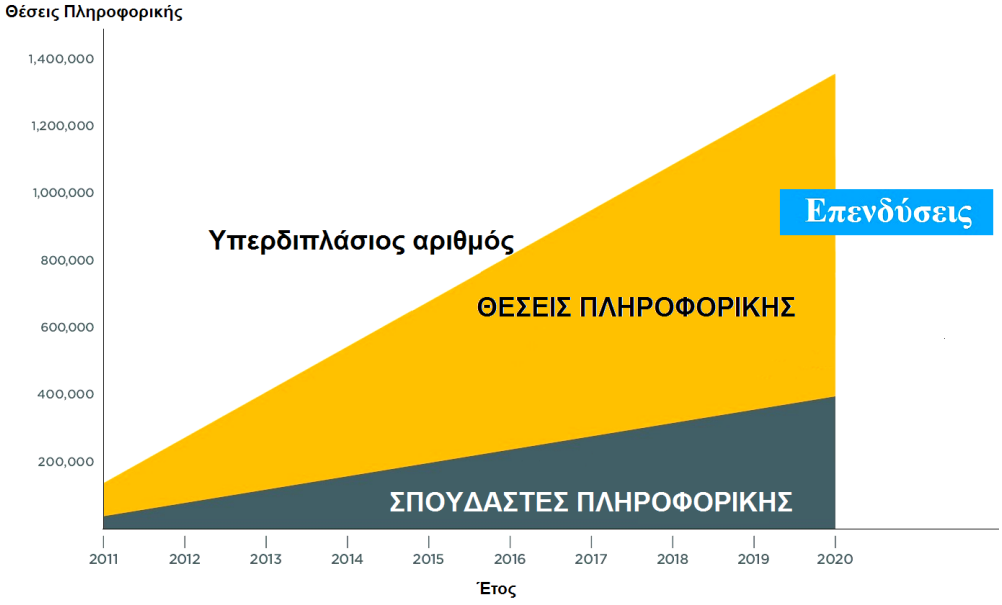
Σύμφωνα με έρευνα από την επιστημονική ομάδα του καθηγητή Θεόδωρου Κατσανέβα, το επάγγελμα του μηχανικού πληροφορικής κατέχει θετικές προοπτικές για τα επόμενα 5 έως 10 χρόνια. Με επένδυση στη γνώση και τη δυνατότητα που δίνει το επάγγελμα του μηχανικού πληροφορικής και γενικά η πληροφορική μέσα από την οποία μπορεί κάποιος να εργάζεται από το σπίτι του με σχεδόν μηδενικό κεφάλαιο μπορεί να δημιουργήσει μια τεραστία επαγγελματική καριέρα και επιτυχία όπως πολλά πρόσωπα στο σήμερα. Για παράδειγμα, η Julieth Brindak (16 χρονών) δημιούργησε το δικό

της κοινωνικό δίκτυο για εφήβους, το οποίο καλείται «Miss O & Friends».

Αποτέλεσμα, σήμερα ο ιστότοπος αποφέρει 15 εκατομμύρια δολάρια! Επίσης ο Nick D'Aloisio (17 χρονών) δημιουργώντας μια εφαρμογή για έξυπνα τηλέφωνα την πούλησε στην εταιρεία Yahoo στο ποσό των 30 εκατομμυρίων δολαρίων! Βέβαια ας μην ξεχνάμε τον ιδρυτή του Facebook καθώς και παλαιότερα τον Bill Gates με τον Paul Allen που ίδρυσαν της Microsoft. Σήμερα σύμφωνα με το γράφημα του Σχήματος 1.4 ο αριθμός θέσεων πληροφορικής ανέρχεται σε υπερδιπλάσιο βαθμό από τον υπάρχοντα αριθμό σπουδαστών της πληροφορικής (πηγή: hour of code).

Στο Σχήμα 1.5 παρουσιάζονται μερικές από τις σημερινές και μελλοντικές εφαρμογές της πληροφορικής.



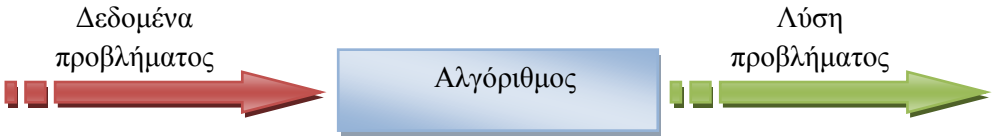


Σχήμα 1.4: Θέσεις πληροφορικής και αριθμός σπουδαστών της πληροφορικής.

Σχήμα 1.5: Σύγχρονες και μελλοντικές εφαρμογές της πληροφορικής.

1.4 Ορισμός αλγόριθμου

Ως αλγόριθμος ορίζεται ένα σύνολο καθορισμένων βημάτων που εκτελούνται σε πεπερασμένο χρόνο και στοχεύουν στην επίλυση ενός προβλήματος. Ο αλγόριθμος γενικότερα είναι μια σύνθετη δομή, η οποία τροφοδοτείται με τα δεδομένα ενός προβλήματος, ακολουθεί τα εκτελέσιμα βήματα που ορίζονται από τη δομή του και παράγει ένα αποτέλεσμα που αντιστοιχεί στη λύση του προβλήματος. Στο Σχήμα 1.6 απεικονίζεται η διαδικασία που μόλις περιγράψαμε.



Σχήμα 1.6: Σχηματική απεικόνιση έννοιας αλγόριθμου.

Για να κατανοήσουμε καλύτερα την έννοια του αλγόριθμου, θα φέρουμε ένα παράδειγμα από την καθημερινή μας ζωή: τη συναρμολόγηση ενός ποδηλάτου. Ορίζουμε την εκφώνηση του προβλήματος:

Πώς θα συναρμολογήσουμε ένα ποδήλατο;

Η διαδικασία που ακολουθούμε είναι να συγκεντρώσουμε όλα τα εξαρτήματα που απαιτούνται για τη συναρμολόγηση ενός ποδηλάτου (σκελετός, ρόδες, γρανάζια, σασμάν ταχυτήτων, αλυσίδα, φρένα, λάστιχα κτλ.), καθώς και τα εργαλεία (κλειδιά, κατσαβίδια κτλ.). Αυτά αποτελούν τα δεδομένα του προβλήματος. Έπειτα πρέπει να ακολουθήσουμε μια ροή ενεργειών, όπως να βιδώσουμε τις ρόδες στον σκελετό, να συνδέσουμε τα γρανάζια στον σκελετό, να περάσουμε την αλυσίδα στα γρανάζια, να βιδώσουμε το τιμόνι και τη σέλα, να περάσουμε τα φρένα κτλ. Αυτές οι ενέργειες αποτελούν τα εκτελέσιμα βήματα του αλγόριθμου.

Εκτελώντας τα βήματα που μόλις αναφέραμε, χρησιμοποιώντας τα δεδομένα του προβλήματος, θα έχουμε ως τελικό προϊόν ένα συναρμολογημένο ποδήλατο, που είναι και η λύση στο πρόβλημα που θέσαμε. Μπορούμε λοιπόν να χαρακτηρίσουμε τα βήματα που ακολουθήσαμε ως έναν αλγόριθμο συναρμολόγησης ενός ποδηλάτου.

Τα βήματα αυτά είναι προκαθορισμένα και ακολουθούν μια συγκεκριμένη σειρά, όμως για τον συγκεκριμένο αλγόριθμο θα μπορούσαμε να αλλάξουμε τη

σειρά των βημάτων, π.χ. να βιδώσουμε πρώτα τα γρανάζια στον σκελετό και μετά τις ρόδες. Αυτό δεν θα επηρέαζε το τελικό αποτέλεσμα, αλλά ίσως επηρέαζε τον χρόνο εκτέλεσης των βημάτων (χρόνος εκτέλεσης αλγόριθμου) μέχρι να παραχθεί το τελικό αποτέλεσμα.

Η λέξη «αλγόριθμος» προέρχεται από μια μελέτη του Πέρση μαθηματικού του 8ου αιώνα μ.Χ. Αλ Χουαρίζμι, η οποία περιείχε συστηματικές τυποποιημένες λύσεις αλγεβρικών προβλημάτων και αποτελεί ίσως την πρώτη πλήρη πραγματεία άλγεβρας. Πέντε αιώνες αργότερα η μελέτη μεταφράστηκε στα λατινικά και άρχιζε με τη φράση «Algorithmus dixit...» («Ο Αλγόριθμος είπε...»). Έτσι η λέξη «αλγόριθμος» καθιερώθηκε αργά τα επόμενα χίλια χρόνια με την έννοια «συστηματική διαδικασία αριθμητικών χειρισμών». Τη σημερινή της σημασία την οφείλει στη γρήγορη ανάπτυξη των ηλεκτρονικών υπολογιστών στα μέσα του 20ού αιώνα. (πηγή: Βικιπαίδεια, http://el.wikipedia.org)

1.5 Αναπαράσταση αλγόριθμου

Ανάλογα με τη φάση υλοποίησης του αλγόριθμου στην οποία βρισκόμαστε, θα μπορούσαμε να πούμε πως υπάρχουν διάφορες μορφές αναπαράστασης. Αν βρισκόμαστε σε διαδικασία μελέτης προκειμένου να υλοποιήσουμε έναν αλγόριθμο, συναντάμε τις ακόλουθες μορφές αναπαράστασης:

Φυσική γλώσσα: Πρόκειται για μια μορφή αναπαράστασης στην οποία περιγράφουμε τη λειτουργία ενός αλγόριθμου σε μορφή ελεύθερου κειμένου χωρίς περιορισμούς στη σύνταξη των εντολών. Ένα παράδειγμα αυτής της αναπαράστασης αναφέρεται στη συνέχεια:

Αλγόριθμος υπολογισμού του μέσου όρου 5 ακέραιων αριθμών

Ο χρήστης εισάγει τον πρώτο ακέραιο αριθμό και αυτός αποθηκεύεται σε μια μεταβλητή με όνομα Χ1. Η διαδικασία επαναλαμβάνεται 5 φορές, έως ότου ο χρήστης έχει εισαγάγει 5 ακέραιους αριθμούς που έχουν αποθηκευτεί στις μεταβλητές Χ1, Χ2, Χ3, Χ4 και Χ5. Αφού ολοκληρωθεί και η εισαγωγή του 5^{ου} αριθμού, το πρόγραμμα προσθέτει τους αριθμούς Χ1+Χ2+Χ3+Χ4+Χ5,

αποθηκεύει το αποτέλεσμα στη μεταβλητή *ΑΘΡΟΙΣΜΑ* (SUM) και διαιρεί την τιμή που βρίσκεται αποθηκευμένη στη μεταβλητή *ΑΘΡΟΙΣΜΑ* με το σύνολο των αριθμών, που είναι 5. Το αποτέλεσμα αποθηκεύεται στη μεταβλητή ΜΟ και εμφανίζεται στον χρήστη.

Αυτού του είδους η αναπαράσταση μπορεί να γίνει μη αποτελεσματική κατά τη φάση της υλοποίησης ενός αλγόριθμου λόγω κακής ποιότητας της περιγραφής που μπορεί να μας δίνει.

Φυσική γλώσσα με βήματα: Πρόκειται για μια μορφή αναπαράστασης που μοιάζει με την αναπαράσταση φυσικής γλώσσας με τη διαφορά πως εδώ ορίζονται συγκεκριμένα βήματα (αριθμημένα), τα οποία περιγράφονται με φυσική γλώσσα. Στη συνέχεια αναφέρουμε ένα παράδειγμα:

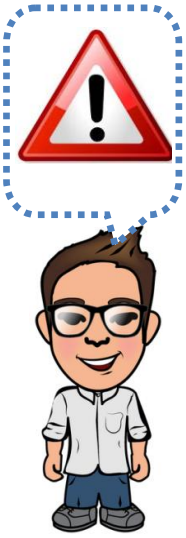
Αλγόριθμος υπολογισμού του μέσου όρου 5 ακέραιων αριθμών

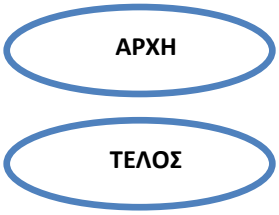
1. Εισαγωγή 1^{ου} ακέραιου αριθμού – αποθήκευση στη μεταβλητή X1
2. Εισαγωγή 2^{ου} ακέραιου αριθμού – αποθήκευση στη μεταβλητή X2
3. Εισαγωγή 3^{ου} ακέραιου αριθμού – αποθήκευση στη μεταβλητή X3
4. Εισαγωγή 4^{ου} ακέραιου αριθμού – αποθήκευση στη μεταβλητή X4
5. Εισαγωγή 5^{ου} ακέραιου αριθμού – αποθήκευση στη μεταβλητή X5
6. Πρόσθεση των μεταβλητών X1…X5 και αποθήκευση στη μεταβλητή *ΑΘΡΟΙΣΜΑ*
7. Διαίρεση της μεταβλητής *ΑΘΡΟΙΣΜΑ* με το 5 και αποθήκευση στη μεταβλητή ΜΟ.
8. Εμφάνιση της τιμής της μεταβλητής ΜΟ στον χρήστη.

Σε αυτή τη μορφή αναπαράστασης η περιγραφή είναι περισσότερο οργανωμένη σε σχέση με την αναπαράσταση απλής φυσικής γλώσσας και η διαδικασία υλοποίησης ενός αλγόριθμου μπορεί να γίνει με μεγαλύτερη ευκολία. Όμως υπάρχει ο κίνδυνος να μην έχουν καθοριστεί σωστά κάποια βήματα και ο αλγόριθμος που θα προκύψει μετά την υλοποίησή του να μην είναι έγκυρος.

Γραφική αναπαράσταση αλγόριθμου (διαγράμματα ροής): Μια πολύ αποτελεσματική μέθοδος αναπαράστασης που χρησιμοποιείται συχνά για την περιγραφή αλγόριθμου στη φάση του σχεδιασμού ή κατά την περιγραφή του, είναι η αναπαράστασή τους με γραφικό τρόπο.

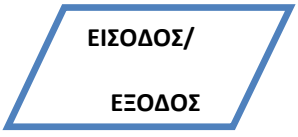
Έχοντας μια γραφική απεικόνιση των λειτουργιών ενός αλγόριθμου και των βημάτων που εκτελούνται κάθε φορά, καθώς και της ροής των δεδομένων μέσα σε αυτόν, είναι πολύ πιο εύκολο να γίνει η υλοποίησή του και η κατανόησή του. Υπάρχουν πολλές γραφικές αναπαραστάσεις αλγόριθμου, με πιο γνωστή ανάμεσά τους το διάγραμμα ροής. Το διάγραμμα ροής είναι μια μέθοδος αναπαράστασης που χρησιμοποιεί γεωμετρικά σχήματα που αντιπροσωπεύουν λειτουργίες και δομές ενός αλγόριθμου, καθώς και βέλη που αντιπροσωπεύουν τη ροή δεδομένων μέσα σε αυτόν. Στη συνέχεια αναλύουμε τα βασικά γεωμετρικά σχήματα που συναντάμε σε ένα διάγραμμα ροής:



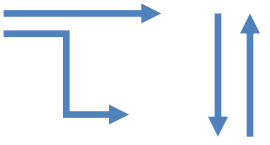


Έλλειψη: Το γεωμετρικό σχήμα της έλλειψης σε ένα διάγραμμα ροής αντιπροσωπεύει την έναρξη ή τη λήξη μιας διαδικασίας ή ενός αλγόριθμου. Όταν πρόκειται για την έναρξη μιας διαδικασίας, μέσα στο σχήμα της έλλειψης αναγράφεται συνήθως η λέξη «έναρξη» ή η λέξη «αρχή» ενώ όταν πρόκειται για τον τερματισμό της αναγράφεται η λέξη «λήξη» ή «τερματισμός» ή «τέλος».

Ορθογώνιο παραλληλόγραμμο: Το γεωμετρικό σχήμα του ορθογωνίου παραλληλογράμμου αντιπροσωπεύει εντολές επεξεργασίας δεδομένων ή διεργασίες. Μέσα σε αυτά αναγράφονται, συνήθως σε συμβολική μορφή, εκτελέσιμες διαδικασίες, όπως εκχώρηση τιμών σε μεταβλητές και εκτέλεση υπολογισμών.



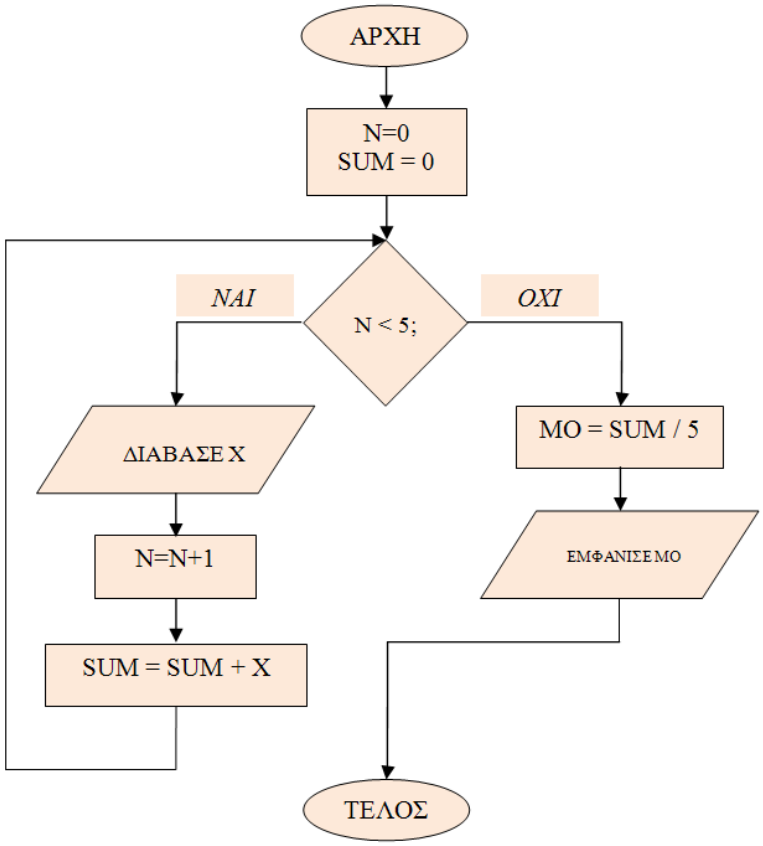
Παραλληλόγραμμο: Το γεωμετρικό σχήμα του παραλληλογράμμου αντιπροσωπεύει διαδικασίες εισόδου και εξόδου δεδομένων ενός αλγόριθμου.



Βέλη: Τα βέλη σε ένα διάγραμμα ροής αντιπροσωπεύουν τη ροή των δεδομένων και των διαδικασιών μέσα σε αυτό. Τα βέλη μπορεί να έχουν διάφορες κατευθύνσεις μέσα σε ένα διάγραμμα ροής.

Ρόμβος: Το γεωμετρικό σχήμα του ρόμβου αντιπροσωπεύει λειτουργίες απόφασης του τύπου αληθές ή ψευδές, σωστό ή λάθος, ισχύει ή δεν ισχύει κτλ. Ο ρόμβος χρησιμοποιείται σε ένα διάγραμμα ροής για να γίνει έλεγχος της ροής των δεδομένων ενός αλγόριθμου ή μιας διαδικασίας μέσω μιας απόφασης. Η ροή των δεδομένων, ανάλογα με την απόφαση, ορίζεται με τη χρήση βέλους από τις άκρες του ρόμβου.

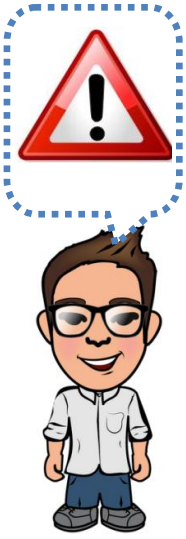
Στο Σχήμα 1.7 απεικονίζεται το διάγραμμα ροής αλγόριθμου υπολογισμού του μέσου όρου 5 ακέραιων αριθμών.



Σχήμα 1.7 Διάγραμμα ροής αλγόριθμου υπολογισμού του μέσου όρου 5 ακέραιων αριθμών.

1.6 Ψευδοκώδικας

Χρησιμοποιώντας το διάγραμμα ροής για να αναπαραστήσουμε έναν αλγόριθμο ή μια διαδικασία, ο βαθμός κατανόησής του μεγαλώνει και γίνεται ευκολότερη η διαδικασία υλοποίησής του, αφού πλέον τα βήματα της εκτέλεσης του απεικονίζονται διαγραμματικά. Το διάγραμμα ροής όμως, όσο κατανοητό και αν φαίνεται, δεν μπορεί να καλύψει όλες τις ανάγκες αναπαράστασης ενός αλγόριθμου προς την τελική υλοποίησή του. Στόχος με τις διάφορες μεθόδους αναπαράστασης αλγόριθμου είναι η όσο το δυνατόν λεπτομερέστερη ανάλυση της λειτουργίας τους, ώστε να εξαχθεί ένα πλάνο στο οποίο θα στηριχθεί μια πρακτική **γλώσσα προγραμματισμού** για την υλοποίησή του αλγόριθμου. Οι μέθοδοι αναπαράστασης που έχουμε αναφέρει μέχρι στιγμής δεν προσεγγίζουν αρκετά μια πραγματική υλοποίηση αλγόριθμου, παρά μόνο περιγράφουν τα βήματα της εκτέλεσής του. Η πραγματική υλοποίηση αλγόριθμου γίνεται με τη χρήση γλωσσών προγραμματισμού. Η μέθοδος αναπαράστασης αλγόριθμου που προσεγγίζει περισσότερο μια πραγματική γλώσσα προγραμματισμού είναι ο ψευδοκώδικας, τον οποίο αναφέρουμε στη συνέχεια.

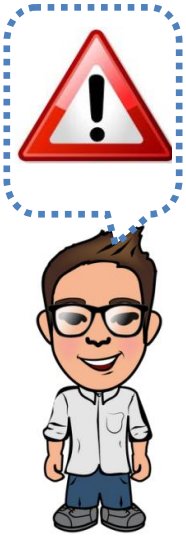


Ψευδοκώδικας: Ο ψευδοκώδικας είναι ένας συνδυασμός αναπαράστασης φυσικής γλώσσας και στοιχείων του δομημένου προγραμματισμού. Οι βασικές αλγοριθμικές δομές που χρησιμοποιούνται στις γλώσσες προγραμματισμού είναι η **ακολουθία**, η **επιλογή** και η **επανάληψη**. Χρησιμοποιώντας έτσι μια ελεύθερη έκφραση της φυσικής γλώσσας και έναν βαθμό της αυστηρότητας του δομημένου προγραμματισμού, δημιουργείται μια ισχυρή μέθοδος αναπαράστασης που προσεγγίζει σε μεγάλο βαθμό μια πραγματική γλώσσα προγραμματισμού. Όπως και σε μια γλώσσα προγραμματισμού, έτσι και στον ψευδοκώδικα υπάρχουν **βασικές εντολές**, οι οποίες εκτελούν συγκεκριμένες λειτουργίες. Υπάρχει επίσης και ένα **συντακτικό**, που ορίζει τον τρόπο με τον οποίο θα πρέπει να συντάσσονται οι εντολές ώστε να αναγνωρίζονται από τους υπολογιστές. Στη συνέχεια παρουσιάζουμε ένα παράδειγμα αναπαράστασης σε μορφή ψευδοκώδικα για τον αλγόριθμο υπολογισμού του μέσου όρου 5 ακέραιων αριθμών.

```
ΠΡΟΓΡΑΜΜΑ ΜΕΣΟΣ_ΟΡΟΣ_5_ΑΚΕΡΑΙΩΝ_ΑΡΙΘΜΩΝ
    ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Χ, ΑΘΡΟΙΣΜΑ, Ν
     ΠΡΑΓΜΑΤΙΚΕΣ: ΜΟ
    ΑΡΧΗ
       Χ ←0
     Ν ←0
     ΑΘΡΟΙΣΜΑ ← 0
     ΟΣΟ Ν < 5 ΕΠΑΝΑΛΑΒΕ
      ΔΙΑΒΑΣΕ Χ
      ΑΘΡΟΙΣΜΑ ← ΑΘΡΟΙΣΜΑ + Χ
      Ν ← Ν + 1
     ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ

     ΜΟ ← ΑΘΡΟΙΣΜΑ/ 5
     ΓΡΑΨΕ ΜΟ
ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Περισσότερα για τις εντολές που χρησιμοποιήσαμε σε αυτό το παράδειγμα θα αναφέρουμε στη συνέχεια αυτού του κεφαλαίου, κατά την ανάλυση των δομών και των δεδομένων που χρησιμοποιούνται στην επιστήμη των υπολογιστών και στις γλώσσες προγραμματισμού.

1.7 Εντολές και δομές αλγόριθμου



Σε αυτή την ενότητα θα χρησιμοποιήσουμε μορφές αναπαράστασης για να μελετήσουμε τις εντολές και τις δομές που χρησιμοποιούνται σε έναν αλγόριθμο. Οι τρεις βασικοί τύποι εντολών που συναντάμε είναι: η εντολή **εισόδου δεδομένων**, η εντολή **εξόδου δεδομένων** και η εντολή **εκχώρησης τιμής** σε μεταβλητή.

Εντολή εισόδου: Είναι μια εντολή που, κατά την εκτέλεσή της, σταματά τη ροή του αλγόριθμου και αναμένει από τον χρήστη να εισάγει δεδομένα προς επεξεργασία από τον αλγόριθμο. Η εισαγωγή των δεδομένων γίνεται συνήθως από το πληκτρολόγιο του υπολογιστή. Μετά το τέλος της πληκτρολόγησης των δεδομένων η ροή εκτέλεσης του αλγόριθμου συνεχίζεται. Τα δεδομένα αποθηκεύονται σε μια μεταβλητή που ορίζουμε κατά τη σύνταξη της εντολής. Η εντολή σε μορφή ψευδοκώδικα συντάσσεται ως εξής:

ΔΙΑΒΑΣΕ Χ;

Όπου Χ είναι η μεταβλητή στην οποία θα αποθηκευτεί η τιμή που θα πληκτρολογήσει ο χρήστης. Ο τύπος δεδομένων αυτής της μεταβλητής μπορεί να είναι κάποιος από τους βασικούς τύπους δεδομένων που αναγράφονται στον ακόλουθο πίνακα:

Τύπος δεδομένων	Τιμές
ΑΚΕΡΑΙΟΣ	Ακέραιες τιμές με ή χωρίς πρόσημο, π.χ. -2, 4, 349, 1459, -9847
ΠΡΑΓΜΑΤΙΚΟΣ	Τιμές με δεκαδικό μέρος με ή χωρίς πρόσημο, π.χ. -989,34 29,3 33,5 -90,32
ΧΑΡΑΚΤΗΡΑΣ	‘Α’, ‘@’, ‘δ’, ‘7’, ‘Ε’, ‘q’
ΔΥΑΔΙΚΟΣ	0 ή 1

Εντολή εξόδου: Είναι μια εντολή που κατά την εκτέλεσή της εμφανίζει μια τιμή στην οθόνη ενός υπολογιστή. Η προς εμφάνιση τιμή βρίσκεται αποθηκευμένη σε μια μεταβλητή η οποία ορίζεται κατά τη σύνταξη της εντολής. Η εντολή σε μορφή ψευδοκώδικα συντάσσεται ως εξής:

ΓΡΑΨΕ Χ;

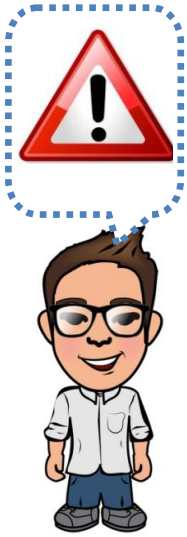
Η μεταβλητή Χ μπορεί να λάβει ακέραιο, πραγματικό, χαρακτήρα ή δυαδικό τύπο δεδομένων.

Εντολή εκχώρησης τιμής: Η εντολή εκχώρησης τιμής κατά την εκτέλεση της αποδίδει απλά μια τιμή σε μια μεταβλητή. Η τιμή αυτή μπορεί να προέρχεται με διάφορες λειτουργίες του αλγόριθμου. Η εντολή συμβολίζεται σε διαφορετικό χαρακτήρα ανάλογα με τη γλώσσα προγραμματισμού, όπως ‘:=’, ‘←’ κ.ά. Η εντολή αυτή σε μορφή ψευδοκώδικα συντάσσεται με διάφορους τρόπους, ως εξής:

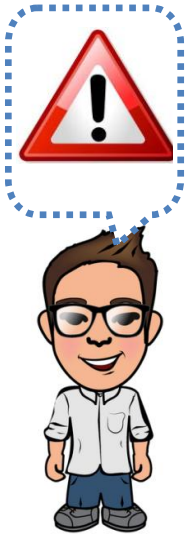
Μορφή εντολής εκχώρησης τιμής	Περιγραφή
X ← 5	Εκχώρηση της τιμής 5 στη μεταβλητή X.
X ← 4 + 5	Εκχώρηση του αθροίσματος 4+5 στη μεταβλητή X.
X ←Y – 5	Εκχώρηση της διαφοράς της τιμής της μεταβλητής Y με τον αριθμό 5 στη μεταβλητή X.
X ←Y + Z	Εκχώρηση του αθροίσματος των τιμών των μεταβλητών Y και Z στη μεταβλητή X.
X ← Z * 4	Εκχώρηση του γινομένου της τιμής της μεταβλητής Z με το 4 στη μεταβλητή X.

Όπως αναφέραμε, οι γλώσσες προγραμματισμού έχουν ένα **συντακτικό**, το οποίο ορίζει με αυστηρούς κανόνες τη σύνταξη των εντολών σε μια γλώσσα, ώστε να αναγνωρίζονται από τους υπολογιστές ως εντολές και όχι ως απλά δεδομένα. Σε αυτό το βιβλίο χρησιμοποιούμε μια μορφή ψευδοκώδικα ώστε να προσεγγίσουμε όσο το δυνατόν καλύτερα μια πραγματική γλώσσα προγραμματισμού. Έτσι ένας από τους βασικούς κανόνες του συντακτικού μας ήταν να ορίσουμε το ελληνικό ερωτηματικό '**;**' ως το σύμβολο τερματισμού μιας εντολής. Με αυτό τον ορισμό, όλες οι εντολές σε οποιονδήποτε κώδικα αλγόριθμου θα τερματίζονται με το σύμβολο '**;**'.

Επίσης το συντακτικό μιας γλώσσας προγραμματισμού ορίζει ένα σύνολο δομών που θα πρέπει να χρησιμοποιούνται κατά τη συγγραφή του κώδικα ενός αλγόριθμου, ώστε να μπορεί να γίνει εκτελέσιμος από έναν ηλεκτρονικό υπολογιστή. Οι θεμελιώδη δομές αλγόριθμου είναι: η **ακολουθία**, η **επιλογή** και η **επανάληψη**.



Η δομή της ακολουθίας: Η δομή αυτή ορίζει πως οι εντολές σε έναν αλγόριθμο εκτελούνται ακολουθιακά η μία μετά την άλλη. Με την ολοκλήρωση της εκτέλεσης μιας εντολής, εκτελείται αμέσως η επόμενη μέχρι το τέλος του αλγόριθμου. Αυτή η δομή χρησιμοποιείται για τη λύση απλών προβλημάτων που απαιτούν την εκτέλεση εντολών με καθορισμένη σειρά. Αυτό όμως δεν είναι πάντοτε πρακτικό για όλα τα προβλήματα. Τα περισσότερα απαιτούν τη λήψη κάποιας απόφασης και τη δυνατότητα επιλογής μεταξύ ενός συνόλου διαφορετικών τιμών προκειμένου να επιλυθούν. Αυτό οδηγεί στην ανάγκη χρήσης δομών επιλογής.



Η δομή της επιλογής: Η δομή αυτή μας δίνει τη δυνατότητα να αποφασίζουμε μέσω μιας συνθήκης, η οποία μπορεί να ισχύει ή όχι, αν θα εκτελεστεί ή όχι αντίστοιχα ένα σύνολο εντολών. Η δομή αυτή σε μορφή ψευδογλώσσας συντάσσεται ως εξής:

```
ΑΝ (συνθήκη) ΤΟΤΕ
   Εντολή 1
   Εντολή 2
    .
    .
    .
      Εντολή ν
ΤΕΛΟΣ_ΑΝ
```

Η δομή αυτή κατά την εκτέλεσή της ελέγχει αν η συνθήκη είναι αληθής (ισχύει) και εκτελεί το σύνολο εντολών που περικλείει. Αν η συνθήκη κατά τον έλεγχό της είναι ψευδής (δεν ισχύει), τότε ο αλγόριθμος συνεχίζει την εκτέλεση των εντολών που βρίσκονται μετά το τέλος της δομής (**ΤΕΛΟΣ_ΑΝ**). Στη συνέχεια παρουσιάζουμε ένα παράδειγμα αλγόριθμου σε μορφή ψευδογλώσσας που χρησιμοποιεί δομή επιλογής για να απόδοση την απόλυτη τιμή της διαφοράς δύο ακέραιων αριθμών:

```
ΠΡΟΓΡΑΜΜΑ ΑΠΟΔΟΣΗ_ΑΠΟΛΥΤΗΣ_ΤΙΜΗΣ_ΔΙΑΦΟΡΑΣ
    ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: X, Y, S

    ΑΡΧΗ
     ΔΙΑΒΑΣΕ X
     ΔΙΑΒΑΣΕ Y

     ΑΝ X >= Y ΤΟΤΕ
      S ← X - Y
     ΤΕΛΟΣ_ΑΝ
```

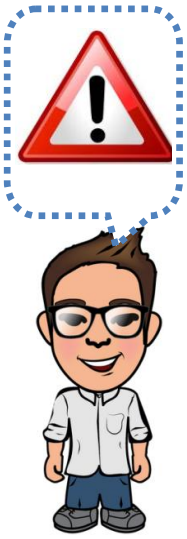
```
    ΑΝ Χ < Υ ΤΟΤΕ
      S ← Y - X
    ΤΕΛΟΣ_ΑΝ

    ΓΡΑΨΕ S
  ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Όπως παρατηρούμε, σε αυτό τον αλγόριθμο χρησιμοποιούμε δύο δομές επιλογής ώστε να ελέγξουμε πότε η τιμή της μεταβλητής Χ είναι μεγαλύτερη ή ίση από την τιμή της μεταβλητής Υ και πότε μικρότερη, ώστε να αφαιρέσουμε με την κατάλληλη σειρά τις τιμές των δύο μεταβλητών για να προκύψει πάντοτε αποτέλεσμα με θετικό πρόσημο (απόλυτη τιμή). Υπάρχει και μια εκδοχή της δομής επιλογής που μας επιτρέπει να κάνουμε πολλαπλές επιλογές με έλεγχο μίας ή περισσότερων συνθηκών. Η δομή αυτή παρουσιάζεται στη συνέχεια σε μορφή ψευδοκώδικα:

```
ΑΝ (συνθήκη 1) ΤΟΤΕ
   Διάφορες εντολές …..
ΑΛΛΙΩΣ ΑΝ (συνθήκη 2) ΤΟΤΕ
      Άλλες εντολές ….
ΑΛΛΙΩΣ ΑΝ (συνθήκη ν) ΤΟΤΕ
       Άλλες εντολές …
ΑΛΛΙΩΣ
      Άλλες εντολές …
ΤΕΛΟΣ_ΑΝ
```

Αυτή η δομή πολλαπλής επιλογής μπορεί να υποστηρίξει τον έλεγχο πολλών συνθηκών προκειμένου να αποδοθεί σε έναν αλγόριθμο η δυνατότητα πολύπλοκων αποφάσεων. Στην πιο απλή εκδοχή της η δομή πολλαπλής επιλογής έχει μόνο μία συνθήκη, η οποία, αν ικανοποιείται, τότε εκτελείται ένα σύνολο εντολών, αλλιώς εκτελείται ένα άλλο σύνολο εντολών. Στη συνέχεια παρουσιάζουμε την απλή εκδοχή της πολλαπλής επιλογής σε μορφή ψευδοκώδικα:

```
ΑΝ (συνθήκη) ΤΟΤΕ
 Εντολές …
ΑΛΛΙΩΣ
          Άλλες εντολές …
ΤΕΛΟΣ_ΑΝ
```



Υπάρχουν περιπτώσεις προβλημάτων που απαιτούν επανάληψη ορισμένων διαδικασιών μέχρι να επιλυθούν. Η λύση τέτοιων προβλημάτων αλγοριθμικά χρησιμοποιεί ***δομές επανάληψης.***

Η δομή της επανάληψης: Η δομή της επανάληψης εκτελεί επαναλαμβανόμενα ένα σύνολο εντολών ελέγχοντας μια συνθήκη που καθορίζει τον τερματισμό ή τη συνέχιση της επανάληψης. Υπάρχουν διάφορες μορφές δομών επανάληψης, που χαρακτηρίζονται από τον τρόπο που ελέγχουν τη συνθήκη τερματισμού τους. Στη συνέχεια αναφέρουμε σε μορφή ψευδογλώσσας τις βασικές δομές επανάληψης:

```
ΟΣΟ (συνθήκη) ΕΠΑΝΑΛΑΒΕ
  Εντολή 1
  Εντολή 2
    …
  Εντολή ν
ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
```

Η δομή επανάληψης **ΟΣΟ** ελέγχει τη συνθήκη τερματισμού και, όσο αυτή είναι αληθής (ισχύει), εκτελεί το σύνολο εντολών που περικλείει. Όταν η συνθήκη τερματισμού πάψει να ισχύει (ψευδής), τότε σταματά η επανάληψη και εκτελούνται οι εντολές, αν υπάρχουν, μετά το **ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ**. Στη συνέχεια αναφέρουμε ένα παράδειγμα αλγόριθμου σε μορφή ψευδοκώδικα που χρησιμοποιεί τη δομή επανάληψης **ΟΣΟ**:

```
ΠΡΟΓΡΑΜΜΑ ΑΘΡΟΙΣΜΑ_30_ΑΚΕΡΑΙΩΝ

  ΜΕΤΑΒΛΗΤΕΣ
    ΑΚΕΡΑΙΕΣ: Χ, Ν, ΑΘΡΟΙΣΜΑ

  ΑΡΧΗ
    ΑΘΡΟΙΣΜΑ ← 0
    Ν←0

    ΟΣΟ Ν < 30 ΕΠΑΝΑΛΑΒΕ
      ΔΙΑΒΑΣΕ Χ
      ΑΘΡΟΙΣΜΑ ←ΑΘΡΟΙΣΜΑ + Χ
      Ν ← Ν + 1
    ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
    ΓΡΑΨΕ ΑΘΡΟΙΣΜΑ
  ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Το παραπάνω πρόγραμμα χρησιμοποιεί τη δομή επανάληψης **ΟΣΟ** για να πραγματοποιήσει 30 φορές τη διαδικασία εισαγωγής ακέραιων τιμών και πρόσθεσης αυτών στη μεταβλητή SUM. Η συνθήκη τερματισμού της δομής **ΟΣΟ** ελέγχει πότε η τιμή της μεταβλητής Ν θα ξεπεράσει τον αριθμό 30. Όσο η τιμή της Ν είναι μικρότερη του 30, η διαδικασία επαναλαμβάνεται και σε κάθε επανάληψη η τιμή της Ν αυξάνεται κατά 1. Έπειτα από 30 επαναλήψεις η συνθήκη τερματισμού θα πάψει να ισχύει (θα γίνει ψευδής) και η δομή επανάληψης θα τερματιστεί, εμφανίζοντας το αποτέλεσμα του αθροίσματος στην οθόνη. Μια παραλλαγή της δομής **ΟΣΟ** είναι η δομή επανάληψης **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ - ΜΕΧΡΙΣ_ΟΤΟΥ**, η οποία επαναλαμβάνει ένα σύνολο εντολών, ελέγχοντας τη συνθήκη τερματισμού της επανάληψης στο τέλος και όχι στην αρχή, όπως η **ΟΣΟ**. Στη συνέχεια παρουσιάζουμε τη δομή αυτή σε μορφή ψευδοκώδικα:

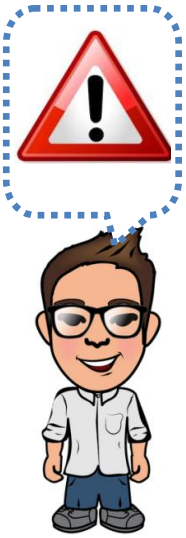
```
ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
  Εντολή 1
  Εντολή 2
    …
  Εντολή ν
ΜΕΧΡΙΣ_ΟΤΟΥ  (συνθήκη)
```

Η ιδιαιτερότητα αυτής της δομής επανάληψης είναι πως θα εκτελέσει το σύνολο εντολών τουλάχιστον μία φορά προτού ελέγξει τη συνθήκη τερματισμού, σε αντίθεση με τη δομή επανάληψης **ΟΣΟ**, η οποία πρώτα ελέγχει τη συνθήκη, με πιθανότητα να μην εκτελέσει καμία φορά τις εντολές που περικλείει. Στη συνέχεια παρουσιάζουμε το παράδειγμα αθροίσματος 30 ακέραιων χρησιμοποιώντας τη δομή **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ - ΜΕΧΡΙΣ_ΟΤΟΥ** αντί της **ΟΣΟ**.

```
ΠΡΟΓΡΑΜΜΑ ΑΘΡΟΙΣΜΑ_30_ΑΚΕΡΑΙΩΝ
    ΜΕΤΑΒΛΗΤΕΣ
     ΑΚΕΡΑΙΕΣ: Χ, Ν, ΑΘΡΟΙΣΜΑ

    ΑΡΧΗ
     ΑΘΡΟΙΣΜΑ ← 0
     Ν ← 0
     ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ
      ΔΙΑΒΑΣΕ Χ
      ΑΘΡΟΙΣΜΑ←ΑΘΡΟΙΣΜΑ + Χ
      Ν ← Ν + 1
     ΜΕΧΡΙΣ_ΟΤΟΥ Ν >= 30
     ΓΡΑΨΕ ΑΘΡΟΙΣΜΑ
    ΤΕΛΟΣ_ΠΡΟΓΡΑΜΜΑΤΟΣ
```

Σε αυτή την εκδοχή του προγράμματος, η συνθήκη τερματισμού της επανάληψης έχει αλλάξει μορφή και επαναλαμβάνουμε τη διαδικασία εισαγωγής ακέραιων αριθμών και πρόσθεσής τους έως ότου η τιμή της μεταβλητής Ν γίνει μεγαλύτερη ή ίση του 30.

Συγκριτικά με την εκδοχή του προγράμματος που χρησιμοποιούσε τη δομή **ΟΣΟ**, η διαφορά βρίσκεται στη σύνταξη της συνθήκης τερματισμού, που στην ουσία έχει απλά αντίθετη λογική. Στα παραδείγματα που έχουμε συναντήσει μέχρι στιγμής και τα οποία αξιοποιούν τις δομές επανάληψης **ΟΣΟ** και **ΑΡΧΗ_ΕΠΑΝΑΛΗΨΗΣ - ΜΕΧΡΙΣ_ΟΤΟΥ**, χρησιμοποιούμε μια ακέραια μεταβλητή Ν ώστε να αυξάνουμε την τιμή της σε κάθε επανάληψη, με αποτέλεσμα, μετά από έναν αριθμό επαναλήψεων, η συνθήκη να τερματιστεί.



Υπάρχει μια μορφή δομής επανάληψης η οποία μας επιτρέπει να πραγματοποιούμε αυτή τη διαδικασία ταυτόχρονα με τον έλεγχο της συνθήκης, χωρίς να είναι απαραίτητο να μεταβάλλουμε την τιμή κάποιας μεταβλητής που επηρεάζει τη συνθήκη τερματισμού μέσα στις εντολές του αλγόριθμου. Στη συνέχεια παρουσιάζουμε τη δομή αυτή σε μορφή ψευδογλώσσας:

```
ΓΙΑ «μεταβλητή» ΑΠΟ …. ΜΕΧΡΙ …. ΜΕ ΒΗΜΑ

  Εντολή 1
  Εντολή 2
     …
  Εντολή ν

ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Η δομή επανάληψης **ΓΙΑ** χρησιμοποιεί μια ακέραια **μεταβλητή** και ορίζει ένα πεδίο τιμών **ΑΠΟ** μια τιμή **ΜΕΧΡΙ** μια άλλη τιμή **ΜΕ_ΒΗΜΑ** αύξησης ή μείωσης και επαναλαμβάνει την εκτέλεση των εντολών που περικλείει όσες φορές ορίζει αυτός ο συνδυασμός. Για να κατανοήσουμε καλύτερα αυτή τη δομή επανάληψης αναφέρουμε το ακόλουθο παράδειγμα ψευδοκώδικα:

```
ΓΙΑ Ν ΑΠΟ 1 ΕΩΣ 30 ΜΕ ΒΗΜΑ 1
 ΔΙΑΒΑΣΕ Χ
 ΑΘΡΟΙΣΜΑ ← ΑΘΡΟΙΣΜΑ + Χ
ΤΕΛΟΣ_ΕΠΑΝΑΛΗΨΗΣ
```

Σε αυτό το παράδειγμα η μεταβλητή Ν θα αυξάνει κατά 1 σε κάθε επανάληψη, ξεκινώντας ΑΠΟ τον αριθμό 1 ΕΩΣ το 30. Σε κάθε επανάληψη εισάγουμε μια ακέραια τιμή και κάνουμε πρόσθεση με την προηγούμενη (απόσπασμα κώδικα από τα προηγούμενα παραδείγματα). Η επανάληψη αυτή θα πραγματοποιηθεί 30 φορές.

2

Υλικό υπολογιστικών συστημάτων

2.1 Ο εσωτερικός κόσμος ενός Η/Υ

Όλοι μας έχουμε χρησιμοποιήσει λίγο πολύ έναν υπολογιστή για να παίξουμε ένα παιχνίδι, να ακούσουμε την αγαπημένη μας μουσική, να γράψουμε μία εργασία σε κάποιο πρόγραμμα επεξεργασίας κειμένου ή ακόμα και να προγραμματίσουμε. Πιάνοντας το ποντίκι και χρησιμοποιώντας το πληκτρολόγιο μπαίνουμε στον κόσμο της τεχνολογίας και ταξιδεύουμε στον κόσμο των πληροφοριών έχοντας σαν πλοηγό τον ηλεκτρονικό μας υπολογιστή. Κάνοντας χρήση του υπολογιστή δεν σκεφτόμαστε συνήθως τον εσωτερικό του κόσμο, πώς δηλαδή αυτός λειτουργεί. Ασχολούμαστε απλά πάνω σε μία οθόνη και σε ένα ποντίκι ή πληκτρολόγιο για να δώσουμε εντολές.

Πώς είναι όμως ο εσωτερικός κόσμος ενός Η/Υ;

Αν ανοίξουμε το κουτί ενός ηλεκτρονικού υπολογιστή μπορούμε να δούμε ένα σύνολο από ηλεκτρονικά εξαρτήματα που βρίσκονται διατεταγμένα πάνω σε μία πλακέτα όπως στο παράδειγμα που απεικονίζεται στο Σχήμα 2.1. Βρισκόμενοι μπροστά από ένα χάος καλωδίων και συσκευών μπορεί το εσωτερικό ενός ηλεκτρονικού υπολογιστή να φαντάζει δύσκολο να κατανοηθεί όμως στην πραγματικότητα δεν είναι και τόσο. Όλοι οι ηλεκτρονικοί υπολογιστές ακολουθούν στην εσωτερική τους δομή μία συγκεκριμένη λογική η οποία αν κατανοηθεί μία φορά για κάποιον υπολογιστή θα μπορεί να κατανοηθεί σχεδόν για όλους του υπολογιστές που κυκλοφορούν στο εμπόριο. Το βασικό για την κατανόηση της εσωτερικής δομής ενός υπολογιστή είναι τα δομικά στοιχεία που τον αποτελούν καθώς και ο τρόπος λειτουργίας τους. Στη συνέχεια θα αναλύσουμε ξεχωριστά τα βασικά δομικά συστατικά ενός ηλεκτρονικού υπολογιστή και έπειτα θα μελετήσουμε τον τρόπο συνδεσμολογίας τους για να δομήσουμε έναν ηλεκτρονικό υπολογιστή.

Σχήμα 2.1: Το εσωτερικό ενός Η/Υ.

2.2 Ο σκληρός δίσκος

Ο σκληρός δίσκος σε έναν ηλεκτρονικό υπολογιστή είναι ένα μέσο αποθήκευσης όπου χρησιμοποιείται για να αποθηκεύει πληροφορίες. Μέσα στον σκληρό δίσκο αποθηκεύεται συνήθως το λειτουργικό σύστημα του υπολογιστή (Windows, Macintosh, Linux κ.α.) και μέσα από αυτό μας δίνεται η δυνατότητα να αποθηκεύουμε προγράμματα (εγκατάσταση), να αποθηκεύουμε έγγραφα, εικόνες, βίντεο και την αγαπημένη μας μουσική. Το σημαντικότερο χαρακτηριστικό του σκληρού δίσκου είναι πως κρατά την πληροφορία αποθηκευμένη ακόμα και όταν δεν τροφοδοτείται με ρεύμα ο ηλεκτρονικός υπολογιστής. Αυτό γίνεται γιατί ο σκληρός δίσκος αποθηκεύει τις πληροφορίες σε μαγνητικό μέσο αφήνοντας ένα μαγνητικό αποτύπωμα πάνω σε αυτό μέσω μίας μαγνητικής βελόνας (Σχήμα 2.2).

Σχήμα 2.2: Εσωτερικό σκληρού δίσκου.

Πως μπορούμε να καταλάβουμε πού ακριβώς βρίσκεται ο σκληρός δίσκος μέσα σε έναν ηλεκτρονικό υπολογιστή; Η απάντηση είναι απλή, είναι το μεγαλύτερο λειτουργικό εξάρτημα συνήθως μέσα σε αυτόν και είναι μεταλλικό κουτί που έχει σχήμα ορθογωνίου παραλληλογράμμου. Επίσης αναγράφει συνήθως την ένδειξη Hard Drive ή HD καθώς και διάφορα χαρακτηριστικά που αυτός έχει όπως, η χωρητικότητά του, η ταχύτητα περιστροφής του δίσκου κ.α.(Σχήμα 2.3).

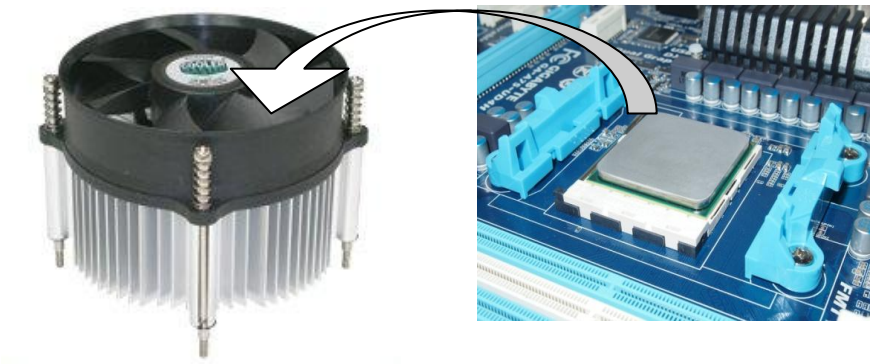
Σχήμα 2.3: Ετικέτα στο πάνω μέρος ενός σκληρού δίσκου.

Η χωρητικότητα του δίσκου είναι επίσης ένα πολύ σημαντικό χαρακτηριστικό και καθορίζει το μέγεθος των πληροφοριών που μπορεί αυτός να αποθηκεύσει. Η χωρητικότητα αποδίδεται σε GB (Gigabytes) και οι σημερινοί δίσκοι έχουν φτάσει σε χωρητικότητα από 160GB (Gigabytes) έως 2 TB (Terabytes). Η ταχύτητα περιστροφής ενός δίσκου μετριέται σε στροφές ανά λεπτό (rpm) και προσδιορίζει την ταχύτητα ανάγνωσης και εγγραφής του δίσκου. Όσο μεγαλύτερη είναι η ταχύτητα περιστροφής τόσο πιο γρήγορη είναι η διαδικασία εγγραφής και ανάγνωσης.

2.3 Ο επεξεργαστής (CPU)

Αν συγκρίνουμε τη δομή ενός ηλεκτρονικού υπολογιστή με την δομή του ανθρώπινου σώματος θα μπορούσαμε να πούμε πως ο επεξεργαστής για τον ηλεκτρονικό υπολογιστή είναι ότι και ο εγκέφαλος για το ανθρώπινο σώμα. Είναι δηλαδή το μέσο με το οποίο ο υπολογιστής σκέπτεται και ελέγχει τα υπόλοιπα δομικά του συστατικά. Έχει ως λειτουργία την εκτέλεση εντολών και το συντονισμό των υπόλοιπων συσκευών μεταξύ τους. Είναι το εξάρτημα που εκτελεί τις περισσότερες λειτουργίες μέσα σε έναν υπολογιστή όμως χωρίς τα υπόλοιπα εξαρτήματα δεν είναι λειτουργικό.

Η μορφή ενός επεξεργαστή απεικονίζεται στο Σχήμα 2.4 όπου μιλάμε για ένα ολοκληρωμένο κύκλωμα. Ανοίγοντας το κουτί ενός ηλεκτρονικού υπολογιστή ο επεξεργαστής κρύβεται σχεδόν πάντοτε πίσω από ένα ανεμιστήρα γι' αυτό και δεν μπορούμε να τον αντικρύσουμε απευθείας. Αν επιθυμούμε να δούμε πως είναι θα πρέπει να αφαιρέσουμε τον ανεμιστήρα (Σχήμα 2.4).



Σχήμα 2.4: Επεξεργαστής και σύστημα ψύξης.

Γιατί όμως χρησιμοποιούμε ανεμιστήρα; Οι επεξεργαστές λόγω της υψηλής συχνότητας λειτουργίας τους και λόγω των πολλών εργασιών που εκτελούν ανεβάζουν τη θερμοκρασία στο ολοκληρωμένο κύκλωμα και προκειμένου να αποφύγουμε κάποια ζημιά σε αυτό χρησιμοποιούμε διάφορα μέσα ψύξης του ολοκληρωμένου κυκλώματος έτσι ώστε αυτό να λειτουργεί σε επιτρεπτές θερμοκρασίες.

Τέτοια μέσα ψύξης είναι οι ψήκτρες και οι ανεμιστήρες επεξεργαστών. Κάποτε η συχνότητα λειτουργίας ενός επεξεργαστή καθόριζε και την ταχύτητά του. Δηλαδή όσο πιο μεγάλη ήταν η συχνότητα λειτουργίας του τόσο πιο γρήγορος ήταν ο επεξεργαστής και βέβαια τόσο πιο πολύ θερμαινόταν και τόσο περισσότερη ενέργεια κατανάλωνε.

Με την πρόοδο όμως της τεχνολογίας και των ολοκληρωμένων κυκλωμάτων καθώς και της επιστήμης της αρχιτεκτονικής επεξεργαστών η ταχύτητα ενός επεξεργαστή καθορίζεται περισσότερο από την αρχιτεκτονική του (τον τρόπο που έχει σχεδιαστεί εσωτερικά) παρά από την συχνότητα λειτουργίας του. Με τις νέες μεθόδους κατασκευής επεξεργαστών οι σημερινοί επεξεργαστές καταναλώνουν λιγότερη ενέργεια, λειτουργούν σε μικρότερες συχνότητες και αποδίδουν περισσότερη επεξεργαστική ισχύ.

2.4 Η μνήμη RAM

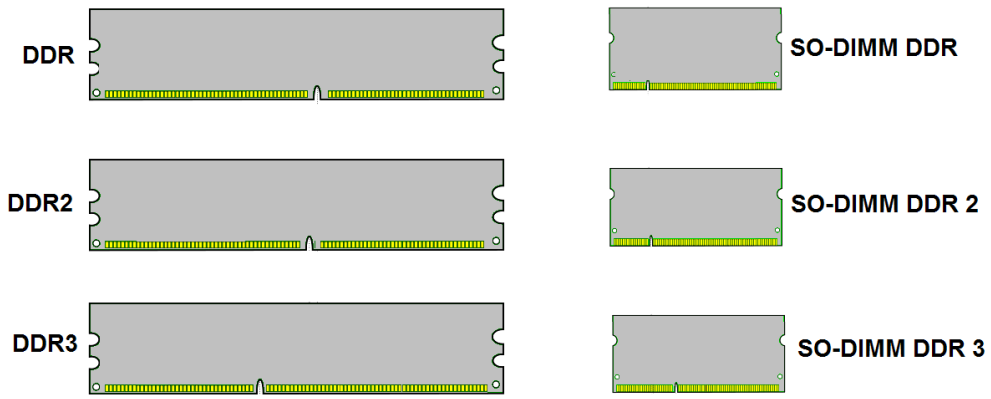
Τα μέσα αποθήκευσης σε έναν ηλεκτρονικό υπολογιστή είναι απαραίτητο να υπάρχουν για ευνόητους λόγους. Ένα μέσο αποθήκευσης που γνωρίσαμε είναι το σκληρός δίσκος. Όπως αναφέραμε ο σκληρός δίσκος μπορεί να αποθηκεύει και να συγκρατεί πληροφορίες με

μαγνητικά μέσα και η ταχύτητά του καθορίζεται από την ταχύτητα περιστροφής του.

Ο σκληρός δίσκος όμως θεωρείται ένα αργό μέσο αποθήκευσης διότι περιέχει μηχανικά μέρη (περιστρεφόμενο δίσκο και μαγνητική βελόνα) που καθυστερούν την ταχύτητα ανάγνωσης και εγγραφής σε αυτόν. Θα έχουμε άλλωστε παρατηρήσει τον χρόνο που λαμβάνει μία διαδικασία αντιγραφής πολλών αρχείων από ένα μέσο αποθήκευσης σε ένα άλλο.

Επειδή λοιπόν ο σκληρός δίσκος είναι ένα όχι και τόσο γρήγορο μέσο αποθήκευσης, για τις βασικές λειτουργίες του υπολογιστή χρησιμοποιείται η μνήμη RAM (Random Access Memory) ή μνήμη τυχαίας προσπέλασης. Αυτή η μνήμη αποθηκεύει προσωρινά όλες τις πληροφορίες που απαιτούνται από ένα λειτουργικό σύστημα ώστε αυτό να λειτουργήσει και μειώνεται σημαντικά η χρήση του σκληρού δίσκου.

Λόγω της κατασκευής της η μνήμη RAM έχει πολύ μεγαλύτερη ταχύτητα προσπέλασης από ότι ο σκληρός δίσκος όμως δεν μπορεί να κρατήσει τα δεδομένα όταν ο ηλεκτρονικός υπολογιστής δεν τροφοδοτείται με ρεύμα. Η τεχνολογία που χρησιμοποιείται για την κατασκευή της μνήμης RAM είναι τεχνολογία ολοκληρωμένων κυκλωμάτων και τα βασικά τους χαρακτηριστικά είναι: η συχνότητα λειτουργίας τους, η χωρητικότητά τους (512 MB, 1GB, 2GB), και ο τύπος τους που προσδιορίζεται από διάφορες τεχνολογίες (DDR, DDR2 ή DDR3) (Σχήμα 2.5).



Σχήμα 2.5: Τύπος μνήμης: α) DDR και β) SO-DIMM.

2.5 Η μητρική πλακέτα (motherboard)

Η μητρική πλακέτα σε έναν ηλεκτρονικό υπολογιστή είναι η βάση πάνω στην οποία συνδέονται όλα τα λειτουργικά μέρη ενός υπολογιστή και μέσω των διαδρόμων που βρίσκονται χαραγμένοι πάνω σε αυτήν μπορούν να επικοινωνούν

και να συντονίζονται μεταξύ τους. Θα μπορούσαμε να πούμε πως η μητρική πλακέτα είναι τα θεμέλια ενός ηλεκτρονικού υπολογιστή. Η μητρική πλακέτα χαρακτηρίζεται ιδιαίτερα από τον τύπο του επεξεργαστή που μπορεί να δεχτεί καθώς και από τον τύπο μνήμης RAM. Πάνω σε αυτή υπάρχουν τοποθετημένες όλες οι θέσεις για να φιλοξενήσουν τον επεξεργαστή, τη μνήμη RAM, τον σκληρό δίσκο, την κάρτα γραφικών καθώς και ένα σύνολο από διάφορα περιφερειακά εξαρτήματα.

Όλες οι θύρες επικοινωνίας του υπολογιστή με τον έξω κόσμο βρίσκονται επίσης ενσωματωμένες πάνω στη μητρική πλακέτα. Η θέση στην οποία τοποθετείται ο επεξεργαστής πάνω στη μητρική πλακέτα ονομάζεται Socket και έχει διαφορετικό σχήμα για κάθε επεξεργαστή. Γι' αυτό λοιπόν όταν επιλέγουμε να αγοράσουμε μία μητρική θα πρέπει να γνωρίζουμε τον επεξεργαστή που επιθυμούμε να χρησιμοποιήσουμε. Ισχύει βέβαια και το αντίθετο. Το ίδιο συμβαίνει και με τη μνήμη RAM όπου υπάρχουν συγκεκριμένες θέσεις που μπορούμε να τις τοποθετήσουμε πάνω στη μητρική πλακέτα και θα πρέπει και αυτές να είναι συμβατές με τον τύπο της υποδοχή (slot) της μνήμης (DDR, DDR2, DDR3).

Επομένως, όπως και με τον επεξεργαστή, θα πρέπει να χρησιμοποιούμε τον τύπο μνήμης που υποστηρίζει η μητρική πλακέτα κάθε φορά. Ας αναλύσουμε όμως τα βασικά στοιχεία που αποτελούν τη μητρική πλακέτα ενός ηλεκτρονικού υπολογιστή.

2.6 Υποδοχή επεξεργαστή (socket)

Ένα από τα βασικά στοιχεία που χαρακτηρίζει μία μητρική πλακέτα είναι η υποδοχή του επεξεργαστή που αυτή υποστηρίζει, το λεγόμενο Socket. Οι μητρικές πλακέτες κατασκευάζονται σύμφωνα με τους διαθέσιμους επεξεργαστές που υπάρχουν στο εμπόριο ακολουθώντας την τεχνολογία που αυτοί υποστηρίζουν.

Το Socket που τοποθετείται ο επεξεργαστής χαρακτηρίζεται από ένα αντιπροσωπευτικό όνομα ή κωδικό ώστε να γνωρίζουν οι τεχνικοί το μοντέλο του επεξεργαστή που μπορεί να δεχτεί η εκάστοτε μητρική πλακέτα. Στο Σχήμα 2.6 απεικονίζονται τα Socket διαφορετικών επεξεργαστών που συναντάμε πάνω σε μητρικές πλακέτες καθώς και τους τύπους επεξεργαστών που υποστηρίζουν.

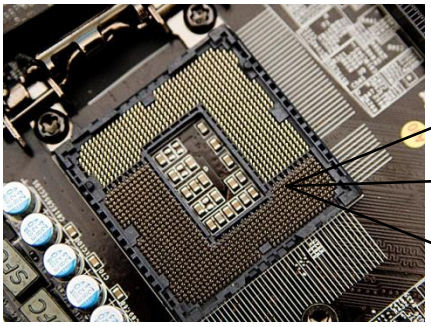
LGA 775

Intel Pentium D

Intel Core 2 Duo

Intel Core 2 Quad

rPGA 988A



Intel Core i7

Intel Core i5

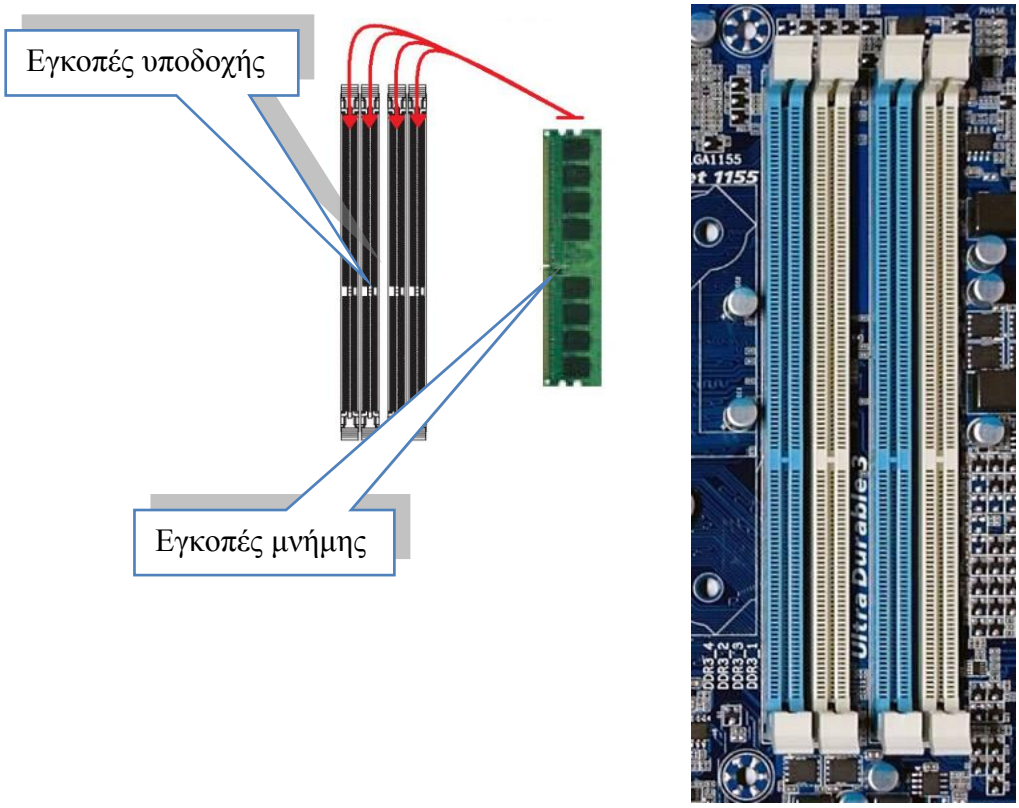
Intel Core i3

Σχήμα 2.6: Τύποι υποδοχών επεξεργαστή.

2.7 Υποδοχές (slots) της μνήμης RAM πάνω στη μητρική

Η μνήμη RAM είναι ένα απαραίτητο στοιχείο για τη λειτουργία του ηλεκτρονικού υπολογιστή και κατέχει συγκεκριμένη θέση πάνω στην μητρική πλακέτα. Ανάλογα με τον τύπο της μνήμης RAM που υποστηρίζει η μητρική πλακέτα έχει και διαφορετικό Slot πάνω σε αυτή. Στο Σχήμα 2.7 απεικονίζονται οι υποδοχές (slots) της μνήμης RAM που συναντάμε πάνω στη μητρική πλακέτα ενός ηλεκτρονικού υπολογιστή.

Με μία πρώτη ματιά στις υποδοχές της μνήμης RAM, παρατηρούμε πως σχεδόν όλες οι μητρικές πλακέτες έχουν το ίδιο σχήμα. Όμως αν κοιτάξουμε πιο προσεκτικά σε διάφορα μοντέλα μητρικών πλακετών θα παρατηρήσουμε πως σε ορισμένα μοντέλα η εγκοπή μέσα στην υποδοχή της μνήμης βρίσκεται σε διαφορετική θέση. Αυτή η εγκοπή προσδιορίζει τον τύπο της μνήμης που μπορεί να δεχτεί η μητρική πλακέτα (Σχήμα 2.7). Επομένως, ανάλογα με τη θέση της εγκοπής στην υποδοχή της μνήμης RAM πάνω στη μητρική καθορίζεται αν η μητρική υποστηρίζει τύπο μνήμης DDR, DDR2 ή DDR3.



Σχήμα 2.7: Υποδοχές (Slots) της μνήμης RAM.

2.8 Σύνθεση ενός ηλεκτρονικού υπολογιστή

Τοποθετώντας τα βασικά εξαρτήματα που γνωρίσαμε πάνω στη μητρική πλακέτα στις υποδοχές που τους αντιστοιχούν, μπορούμε εύκολα να συνθέσουμε την κεντρική μονάδα ενός ηλεκτρονικού υπολογιστή. Στο Σχήμα 2.8 απεικονίζεται μία

ολοκληρωμένη σύνθεση της κεντρικής μονάδας ενός ηλεκτρονικού υπολογιστή.

Σχήμα 2.8: Σύνθεση Η/Υ.

2.9 Νέες τεχνολογίες υπολογιστών

Τα τελευταία χρόνια με την εξέλιξη της τεχνολογίας των υπολογιστών τα πράγματα έγιναν πολύ μικρότερα και πιο προσιτά στον χρήστη. Με την χρήση οθονών αφής και την ανάπτυξη των λειτουργικών συστημάτων, τα υπολογιστικά συστήματα άλλαξαν μορφή και χρησιμοποιούνται περισσότερο ως φορητά συστήματα πλέον και όχι ως μεγάλες μονάδες που στέκονται σταθερές πάνω σε ένα γραφείο με την οθόνη τους, το πληκτρολόγιο και το ποντίκι τους. Την πρώτη επανάσταση την έφεραν τα Laptops και τα Notebooks όπου έκαναν το πρώτο βήμα στον κόσμο της φορητής πληροφόρησης.

Ένα μεγάλο μέρος των χρηστών Η/Υ παγκοσμίως χρησιμοποιούν ως βασικό υπολογιστικό σύστημα των φορητό υπολογιστή (Laptop) ενώ τα 3 τελευταία χρόνια έχει αυξηθεί δραματικά η χρήση των ταμπλετών (Tablets) και των έξυπνων τηλεφώνων (Smartphones) μετά την επανάσταση που έφερε η οθόνη αφής. (Σχήμα 2.9).

Σχήμα 2.9: Σύγχρονα υπολογιστικά συστήματα.

3

Τα πρώτα μου βήματα με το Getcoding

3.1 Το παιχνίδι του Getcoding

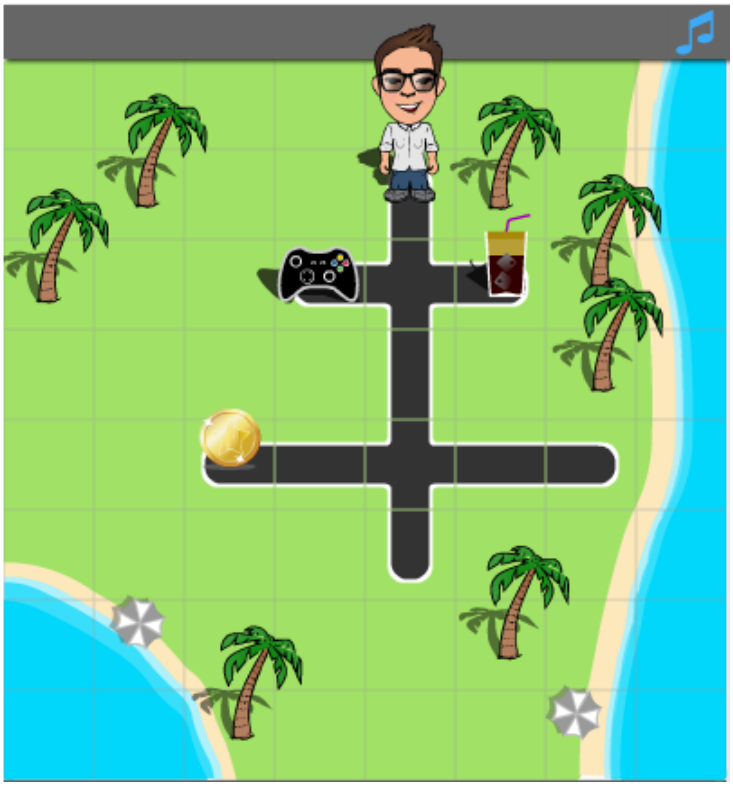
Μέσα από το παιχνίδι του Getcoding μαθαίνουμε τις βασικές έννοιες του προγραμματισμού. Είναι εύκολο και δεν χρειάζονται ειδικές γνώσεις! Σκοπός του προγραμματιστικού παιχνιδιού είναι να οδηγήσουμε την Ίρις και τον Θαλή να πιάσουν το χρυσό νόμισμα γνωρίζοντας και κατανοώντας με τον τρόπο αυτό τα διαγράμματα ροής, τους βασικούς αλγόριθμους, τις επαναλήψεις και τις υποθέσεις.

Σχετικά με τα ονόματα που δόθηκαν στους ήρωες μας, ο Θαλής (640 ή 624 π.Χ. έως 546 π.Χ.) είναι ο αρχαιότερος προσωκρατικός φιλόσοφος, ο πρώτος των επτά σοφών της αρχαιότητας, μαθηματικός, φυσικός, αστρονόμος, μηχανικός, και μετεωρολόγος.

Η Ίρις ήταν κατά την ελληνική μυθολογία δευτερεύουσα θεότητα του Ολύμπου, αναφερόμενη και ως μία από τις Άρπυιες. Ανήκε στην ακολουθία των θεών με καθήκοντα αγγελιαφόρου όμοια με εκείνα του Ερμή.

3.2 Η δομή της ακολουθίας

Σε ένα πρόγραμμα μία πολύ σημαντική δομή είναι η δομή της ακολουθίας. Σύμφωνα με αυτή τη δομή οι εντολές σε ένα πρόγραμμα εκτελούνται η μία μετά την άλλη «ακολουθιακά». Ας γνωρίσουμε καλύτερα τη δομή της ακολουθίας μέσα από το παιχνίδι του Getcoding (Σχ. 3.1).



Σχήμα 3.1: Εικόνα παιχνιδιού.

Σε αυτό το παιχνίδι θα πρέπει να προγραμματίσουμε τη διαδρομή που θα κάνει ο Θαλής για να φτάσει το νόμισμα και να περάσει στην επόμενη πίστα. Σε αυτό το πρόγραμμα ο Θαλής μπορεί να προχωράει ένα βήμα μπροστά, να στρίβει αριστερά ή δεξιά. Τα βήματα που επιτρέπεται να προχωρήσει είναι όσα και τα τετράγωνα που σχηματίζουν ένα πλέγμα στην οθόνη. Παρατηρώντας το μονοπάτι προς το νόμισμα θα δούμε και άλλους πειρασμούς όπως ηλεκτρονικό παιχνίδι και καφές, όμως στόχος είναι να προγραμματίσουμε το παιχνίδι ώστε ο Θαλής να φτάσει στο νόμισμα.

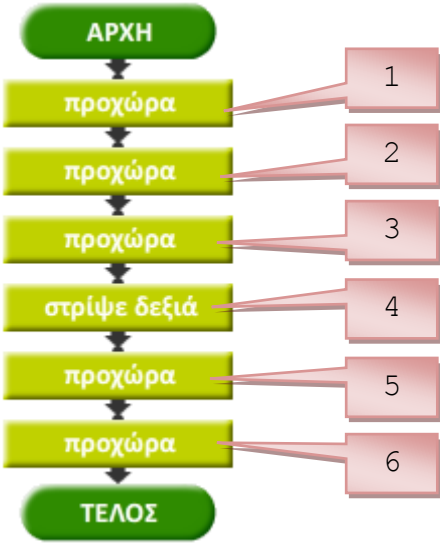
Ας πάρουμε ένα μολύβι και ας αριθμήσουμε τα βήματα που πρέπει να κάνει ο Θαλής μέχρι το νόμισμα. Στο Σχήμα 3.2 έχουμε βάλει αριθμούς στα τετράγωνα

της σωστής διαδρομής. Έχουμε αριθμήσει τα βήματα και τις κινήσεις που πρέπει να κάνει ο Θαλής και όπως βλέπουμε θα πρέπει να κάνει τρία βήματα εμπρός, να στρίψει δεξιά και έπειτα δύο βήματα εμπρός για να φτάσει το νόμισμα.

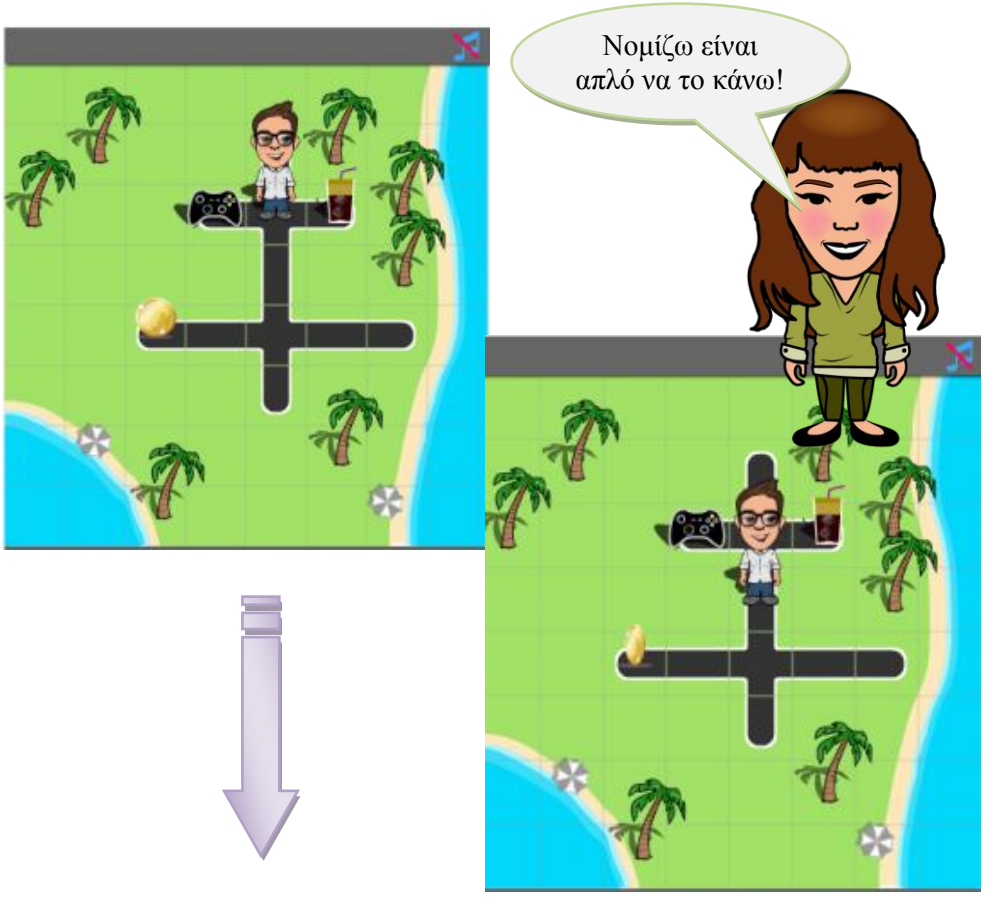
Σχήμα 3.2: Εικόνα παιχνιδιού.

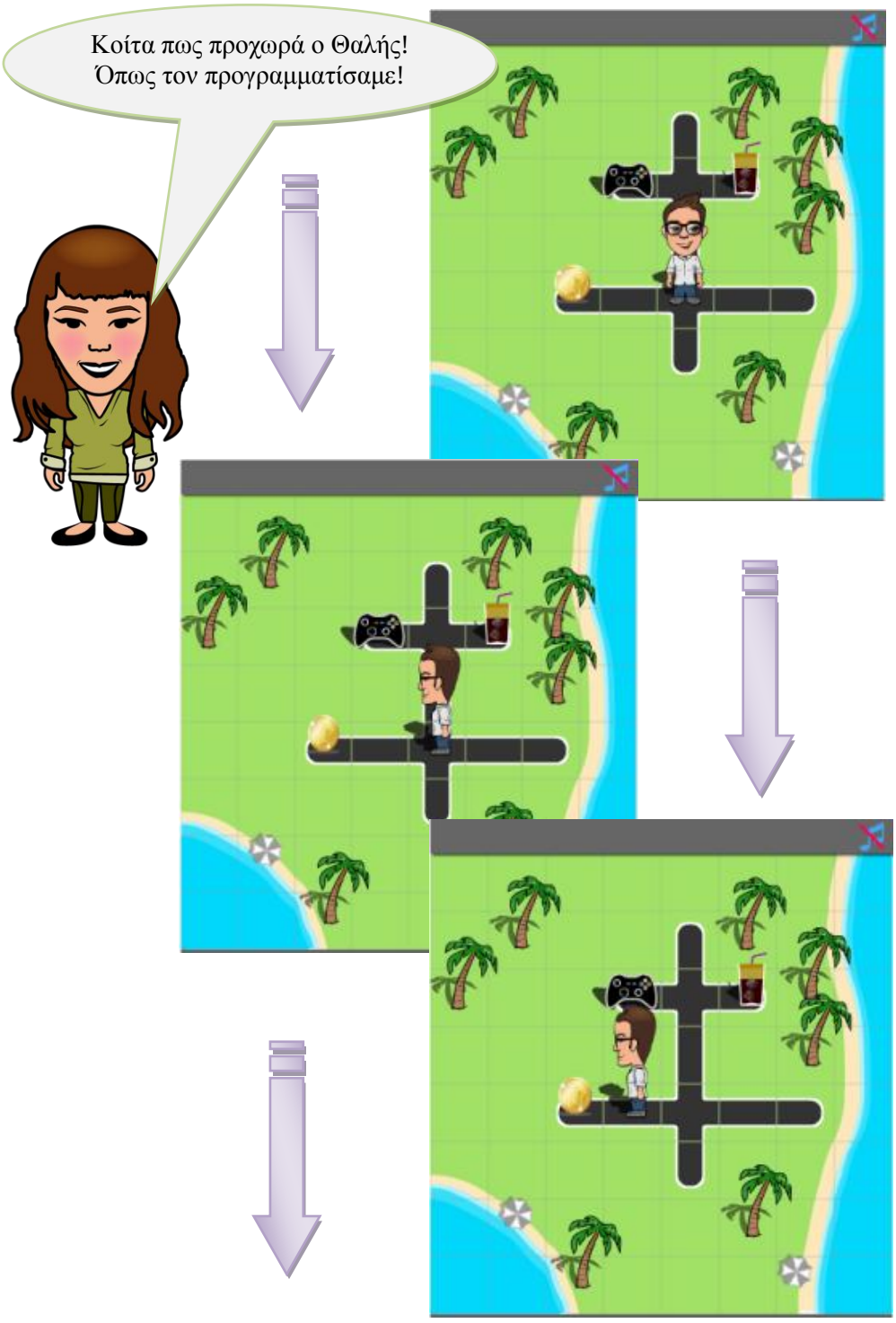
Για να προγραμματίσουμε αυτό το παιχνίδι θα χρησιμοποιήσουμε τη δομή της ακολουθίας και θα δώσουμε εντολές στον Θαλή. Οι εντολές που έχουμε στη διάθεσή μας είναι οι ακόλουθες:

Επομένως, σύμφωνα με τη παραπάνω εικόνα, θα πρέπει να χρησιμοποιήσουμε τρεις εντολές «προχώρα», μία εντολή «στρίψε δεξιά» και δύο ακόμα εντολές «προχώρα» για να οδηγήσουμε τον Θαλή στο νόμισμα. Το πρόγραμμα (Σχ. 3.3) καθώς εκτελείται παρουσιάζεται στη συνέχεια στο Σχήμα 3.4.



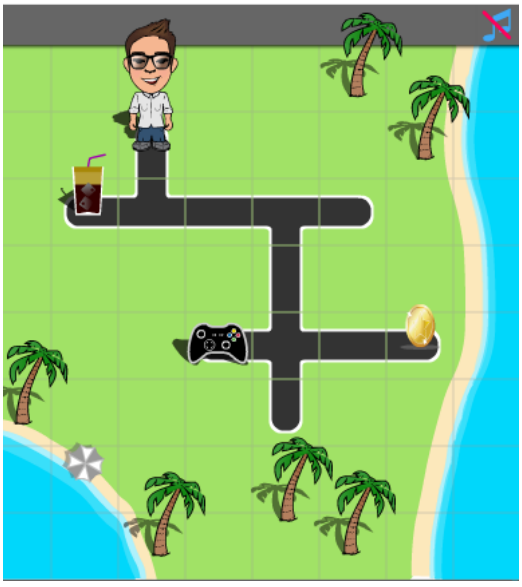
Σχήμα 3.3: Διάγραμμα ροής του κώδικα.




Κοίτα πως προχωρά ο Θαλής!
Όπως τον προγραμματίσαμε!

Σχήμα 3.4: Εικόνα παιχνιδιού.

Ας κάνουμε λίγο πιο δύσκολη τη διαδρομή προς το νόμισμα και ας προσπαθήσουμε να χρησιμοποιήσουμε τη δομή της ακολουθίας με περισσότερες εντολές αυτή τη φορά στην πίστα του Σχήματος 3.5.



Σχήμα 3.5: Εικόνα παιχνιδιού.

Σε αυτή την πίστα θα πρέπει να χρησιμοποιήσουμε περισσότερες εντολές ακολουθιακά γιατί το μονοπάτι προς το νόμισμα είναι μεγαλύτερο. Αλλάζει όμως και η διαδρομή προς το νόμισμα και πρέπει να βρούμε τη νέα διαδρομή η οποία παρουσιάζεται στη συνέχεια (Σχ. 3.6). Έτσι, τα βήματα που πρέπει να κάνει ο Θαλής είναι:

Σχήμα 3.6: Εικόνα παιχνιδιού.

1. Να προχωρήσει μία φορά μπροστά
2. Να στρίψει αριστερά
3. Να προχωρήσει δύο φορές μπροστά
4. Να στρίψει δεξιά
5. Να προχωρήσει δύο φορές μπροστά
6. Να στρίψει αριστερά
7. Να προχωρήσει δύο φορές μπροστά

Το πρόγραμμα λοιπόν έχει περισσότερες εντολές και χρησιμοποιώντας τη δομή της ακολουθίας και τις εντολές που έχουμε μάθει μέχρι τώρα, γράφεται όπως στο διπλανό Σχήμα 3.7.

Σχήμα 3.7: Διάγραμμα ροής του κώδικα.

Η δομή της ακολουθίας χρησιμοποιείται σχεδόν σε όλα τα προγράμματα που γράφουμε όμως πολλές φορές για να λύσουμε ένα πρόβλημα, σαν αυτό του Θαλή με το νόμισμα, χρειάζεται να εκτελέσουμε πολλές εντολές ακολουθιακά μέχρι να λυθεί το πρόβλημά μας. Στα προβλήματα που έχουμε προγραμματίσει μέχρι τώρα ο Θαλής έχει να προχωρήσει περίπου 10 τετράγωνα μέχρι να φτάσει στο νόμισμα και εμείς γράψαμε μέχρι 10 εντολές ακολουθιακά για να οδηγήσουμε τον Θαλή στο νόμισμα. Τι γίνεται όμως αν ο Θαλής έχει να ταξιδέψει 100 ή 1000 τετράγωνα μέχρι το νόμισμα; Με τη δομή της ακολουθίας θα πρέπει να γράψουμε 1000 εντολές κάτι που θα μας πάρει πάρα πολύ καιρό για να το πραγματοποιήσουμε. Για το λόγο αυτό, υπάρχει μία άλλη πολύ σημαντική προγραμματιστική δομή, η δομή της επανάληψης.

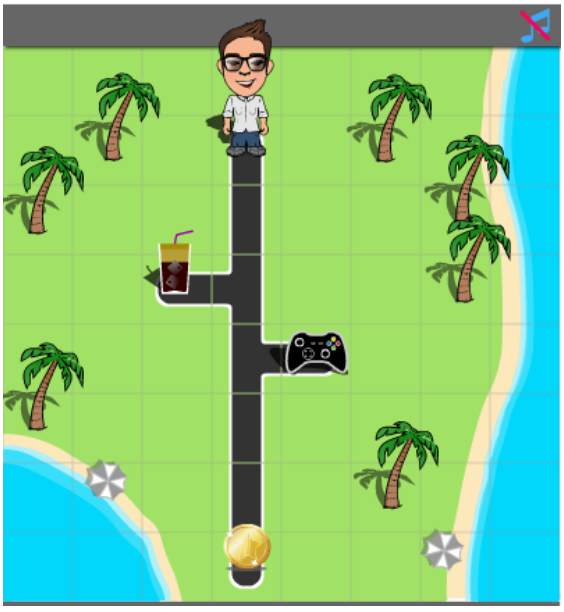
3.3 Η δομή της επανάληψης

Η δομή της επανάληψης χρησιμοποιείται στα προγράμματα για να επαναλαμβάνουμε πολλές φορές ένα τμήμα κώδικα. Για να χρησιμοποιήσουμε μία επανάληψη στον προγραμματισμό θα πρέπει να γνωρίζουμε δύο πράγματα:

1. **Πόσες φορές θέλουμε να επαναλάβουμε κάτι;**

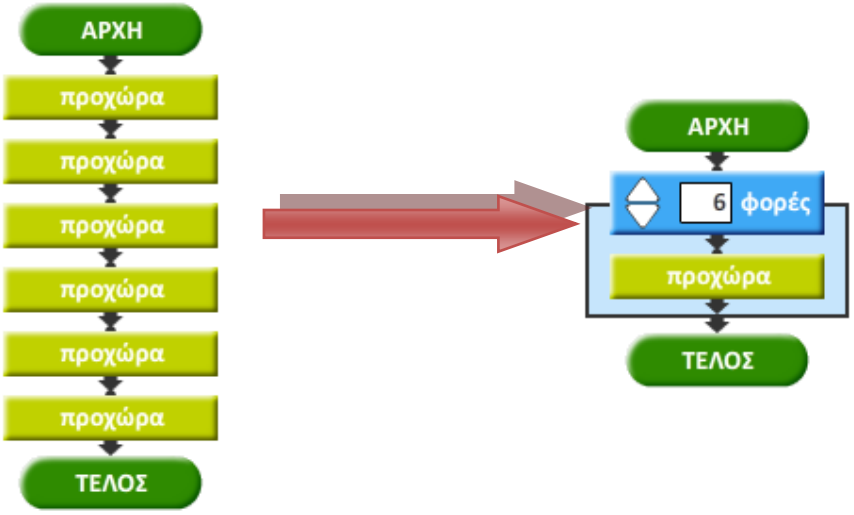
2. **Τι θέλουμε να επαναλάβουμε;**

Στο παράδειγμα της εικόνας του Σχήματος 3.8 θα προσπαθήσουμε να οδηγήσουμε και πάλι τον Θαλή στο νόμισμα. Όπως παρατηρούμε η διαδρομή είναι πολύ εύκολη και θα πρέπει να προχωρήσουμε απλά ευθεία μέχρι το νόμισμα. Μία λύση αυτού του προβλήματος με πρόγραμμα θα ήταν να χρησιμοποιήσουμε 6 εντολές «προχώρα» με τη δομή της ακολουθίας για να οδηγήσουμε τον Θαλή στο νόμισμα. Η λύση αυτή παρουσιάζεται στο Σχήμα 3.9α. Όπως παρατηρούμε το πρόγραμμά μας είναι αρκετά μεγάλο και χρησιμοποιούμε διαρκώς την ίδια εντολή.



Σχήμα 3.8: Εικόνα παιχνιδιού.

Επειδή επαναλαμβάνουμε διαρκώς την ίδια διαδικασία θα ήταν προτιμότερο να χρησιμοποιήσουμε μία δομή επανάληψης και μέσα σε αυτήν να τοποθετήσουμε την εντολή ή τις εντολές που επιθυμούμε να επαναλάβουμε αφού γνωρίζουμε τον αριθμό των επαναλήψεων. Επομένως το πρόγραμμα με δομή επανάληψης θα είναι όπως αυτό που παρουσιάζεται στη εικόνα του Σχήματος 3.9(β).

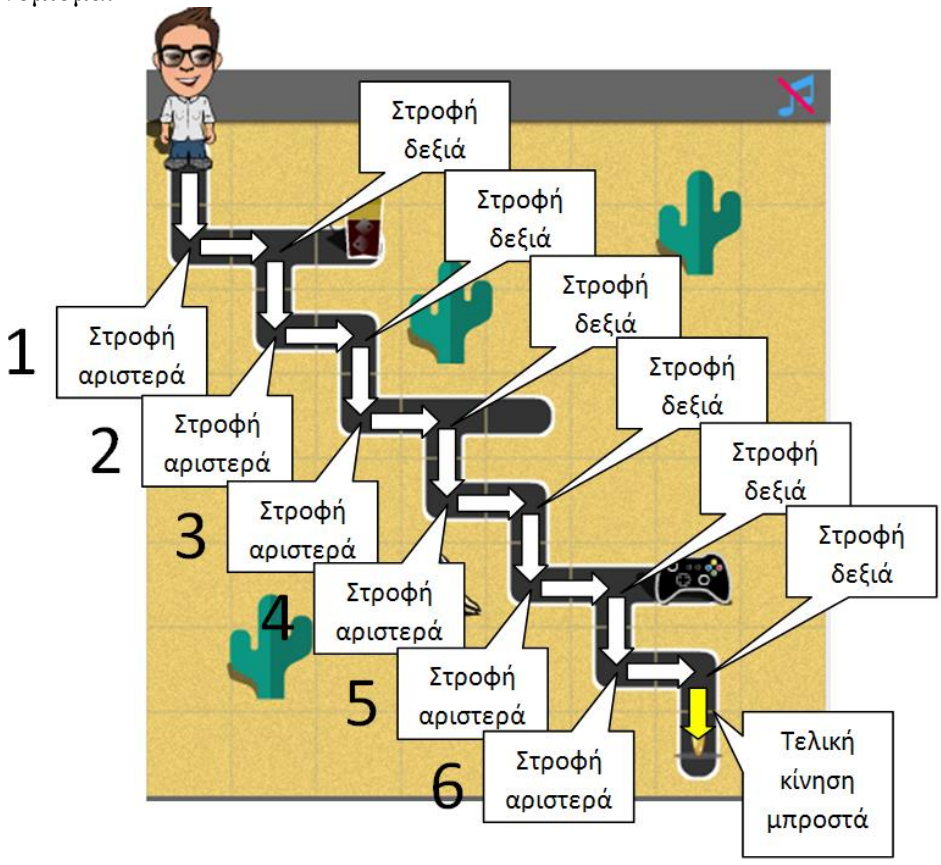


Σχήμα 3.9: α) Διάγραμμα ροής του κώδικα και β) δομή επανάληψης.

Ένα παράδειγμα χρήσης της δομής επανάληψης στον προγραμματισμό είναι στο παιχνίδι που παρουσιάζεται στο Σχήμα 3.10. Ο Θαλής θα πρέπει να φτάσει στον τερματισμό και να πάρει το νόμισμα για να προχωρήσει στην επόμενη πίστα.

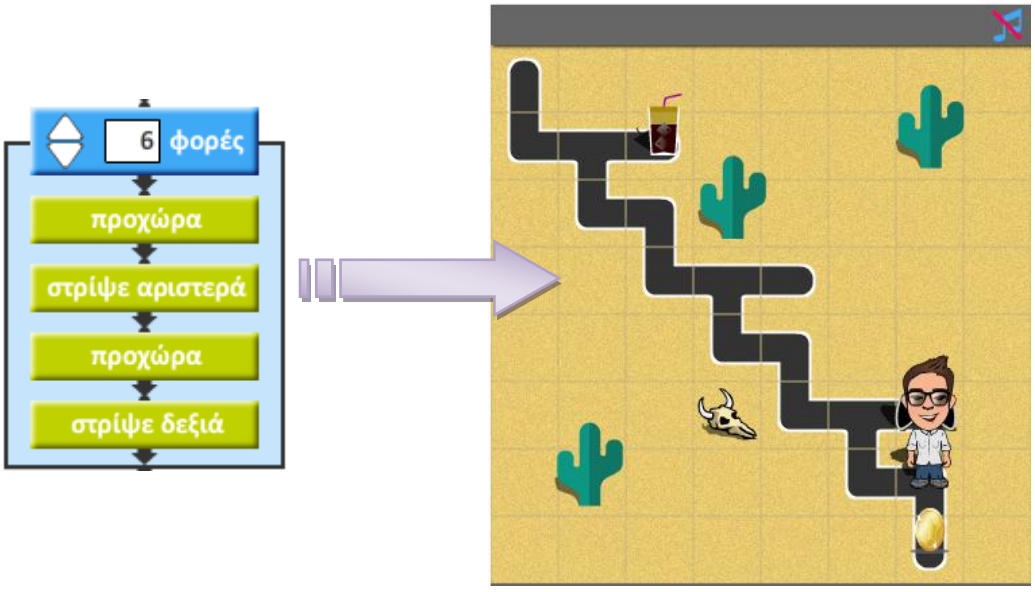
Σχήμα 3.10: Εικόνα παιχνιδιού.

Πρέπει όμως να ακολουθήσει μία διαδρομή προχωρώντας και στρίβοντας δεξιά ή αριστερά ακολουθώντας το μονοπάτι. Υπάρχουν πάλι διάφοροι πειρασμοί στο δρόμο για το νόμισμα, όμως ο προγραμματιστής θα πρέπει να προγραμματίσει το παιχνίδι ώστε να ακολουθήσει τη σωστή διαδρομή. Ο Θαλής μπορεί να προχωρά ένα-ένα τετράγωνο στο πλέγμα της οθόνης όπως έχουμε εξηγήσει και στα προηγούμενα παραδείγματα (Σχ. 3.11). Ας πάρουμε ένα μολύβι και ας ζωγραφίσουμε τη σωστή διαδρομή που πρέπει να κάνει ο Θαλής μέχρι να φτάσει στο νόμισμα!



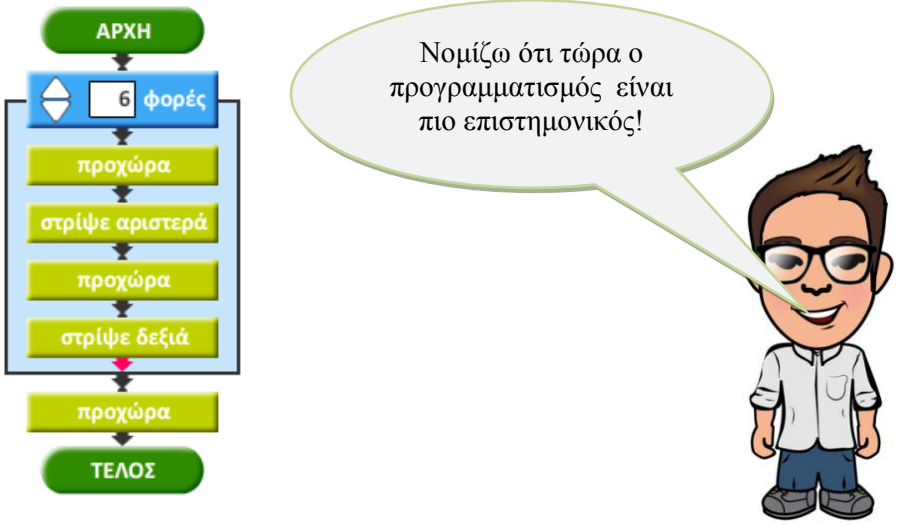
Σχήμα 3.11: Εικόνα παιχνιδιού.

Όπως παρατηρούμε από την σημειωμένη εικόνα ο Θαλής θα πρέπει να επαναλάβει 6 φορές ένα συγκεκριμένο σύνολο κινήσεων όπως αυτό στη διπλανή εικόνα. Αν επαναλάβει αυτές τις κινήσεις 6 φορές ακριβώς θα βρεθεί ένα βήμα πριν το νόμισμα (Σχ. 3.12).



Σχήμα 3.12: Διάγραμμα ροής του κώδικα και εικόνα του παιχνιδιού.

Αφού φτάσουμε ένα βήμα πριν το νόμισμα μπορούμε να εκτελέσουμε μία κίνηση «προχώρα» και να πάρουμε το νόμισμα για να προχωρήσουμε στην επόμενη πίστα. Το πρόγραμμά μας λοιπόν θα είναι όπως αυτό που παρουσιάζεται στη συνέχεια (Σχ. 3.13):

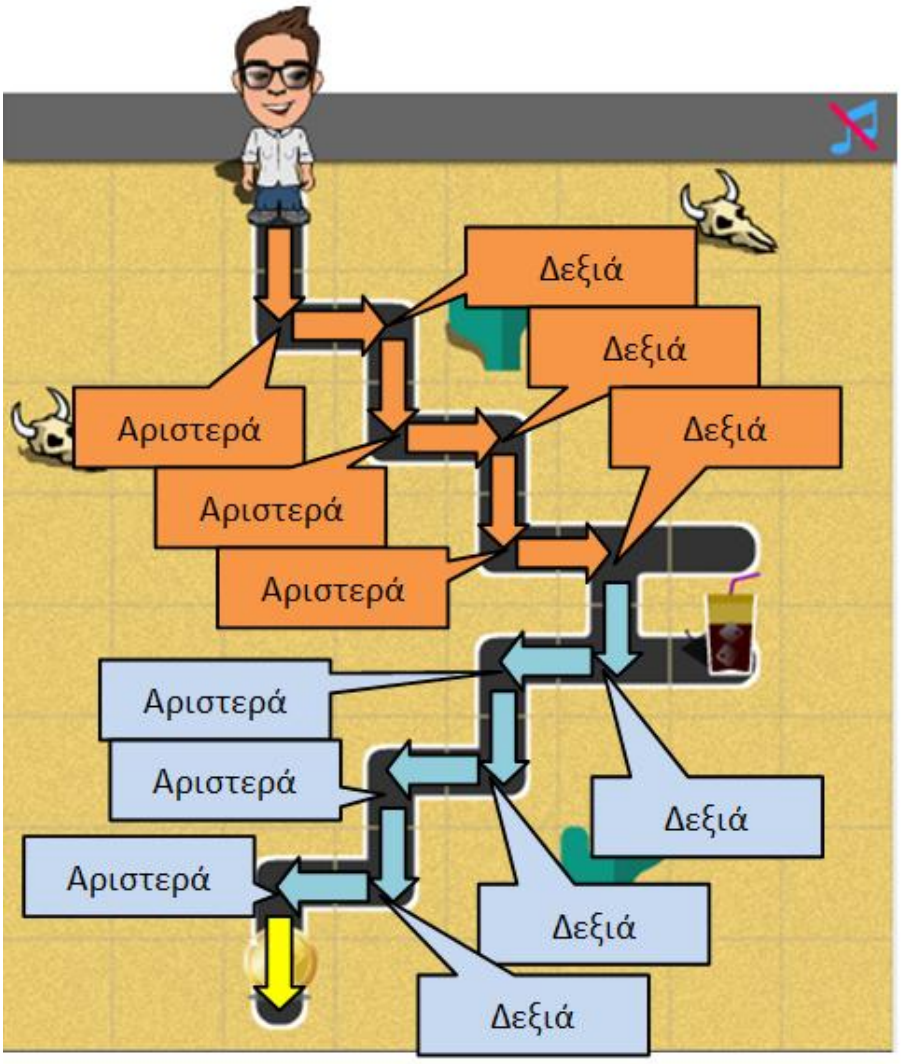


Σχήμα 3.13: Διάγραμμα ροής του κώδικα.

Η δομή της επανάληψης μας βοηθάει να γράφουμε ένα τμήμα κώδικα μόνο μία φορά και να το επαναλαμβάνουμε όσες φορές επιθυμούμε ώστε να λύσουμε ένα πρόβλημα. Χρησιμοποιώντας αυτή τη δομή μπορούμε να μειώσουμε πολύ τον αριθμό των εντολών που γράφουμε σε ένα πρόγραμμα καθώς και τον χρόνο που χρειάζεται για να γράψουμε ένα πρόγραμμα.

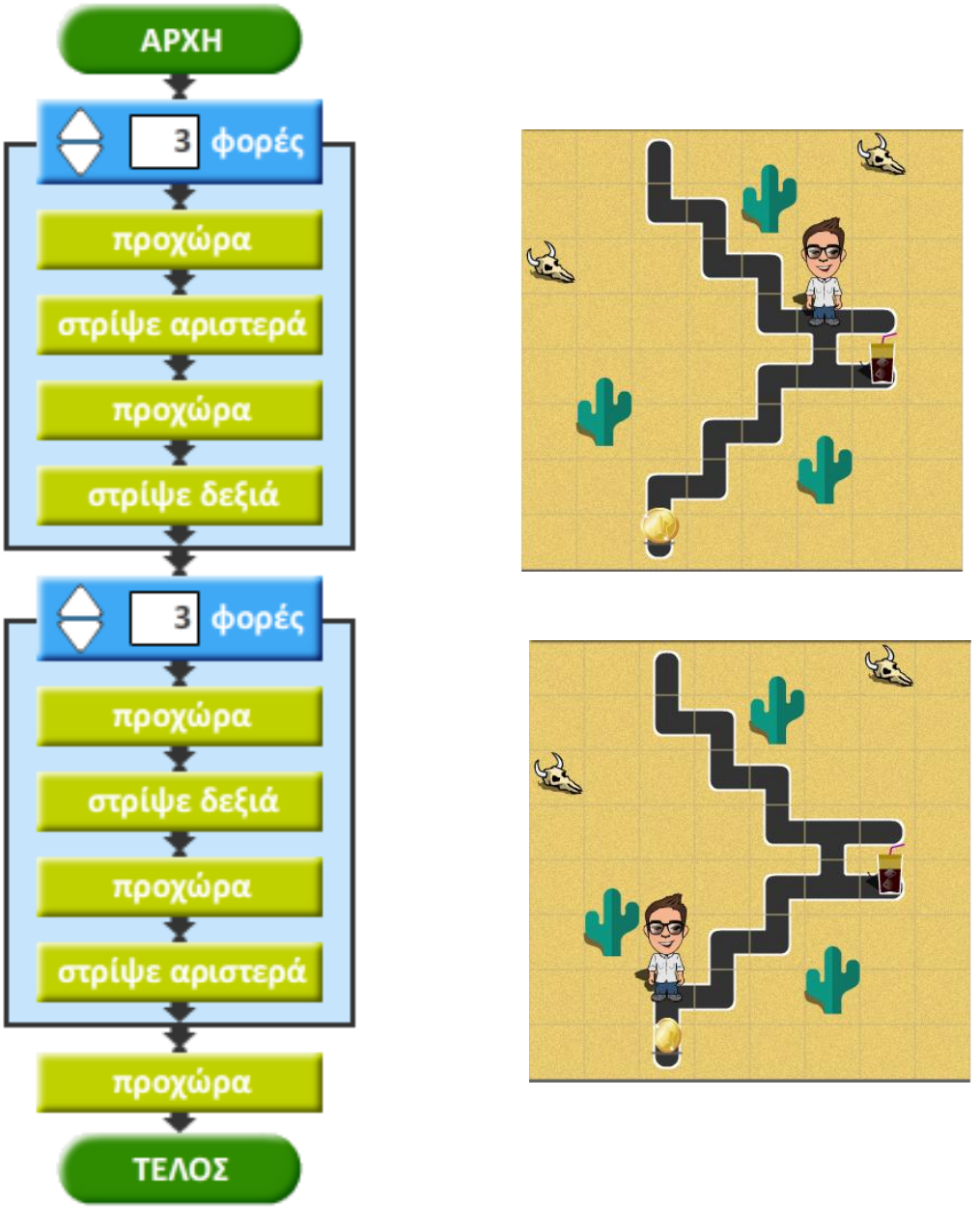
Συνδυάζοντας τη δομή της ακολουθίας και τη δομή της επανάληψης μπορούμε να γράψουμε προγράμματα που λύνουν πολύ γρήγορα ένα μεγάλο αριθμό προβλημάτων. Η δομή της επανάληψης μπορεί να χρησιμοποιηθεί περισσότερες από μία φορές μέσα σε ένα πρόγραμμα αν αυτό χρειάζεται. Πολλά προβλήματα για να λυθούν χρειάζεται να επαναλάβουν αρκετές φορές δύο ή περισσότερες διαφορετικές διαδικασίες. Ας δούμε ένα τέτοιο παράδειγμα με τον Θαλή και το νόμισμα όπου αυτή τη φορά ο Θαλής πρέπει να ακολουθήσει ένα μοτίβο διαδρομής που στην πορεία αλλάζει σε ένα διαφορετικό μοτίβο μέχρι να φτάσει στο νόμισμα (Σχ. 3.14). Στην εικόνα του Σχήματος 3.15 έχουμε σημειώσει τη διαδρομή που πρέπει να κάνει ο Θαλής μέχρι το νόμισμα.

Σχήμα 3.14: Εικόνα παιχνιδιού.



Σχήμα 3.15: Εικόνα παιχνιδιού.

Τα σκούρα γκρι βελάκια δείχνουν το μοτίβο που πρέπει να ακολουθήσουμε μέχρι τη μέση της διαδρομής ενώ τα ανοιχτά γκρι βελάκια δείχνουν το μοτίβο που πρέπει να ακολουθήσουμε μέχρι ένα βήμα πριν το τέλος της διαδρομής προς το νόμισμα. Μπορούμε να χρησιμοποιήσουμε δύο δομές επανάληψης ακολουθιακά ώστε να επαναλάβουμε τα δύο μοτίβα που σχεδιάσαμε. Έπειτα θα εκτελέσουμε μία εντολή «προχώρα» για να φτάσουμε στο νόμισμα. Ο κώδικας του προγράμματος θα είναι ο ακόλουθος (Σχ. 3.16).



Σχήμα 3.16: Διάγραμμά ροής του κώδικα και εικόνα παιχνιδιού.

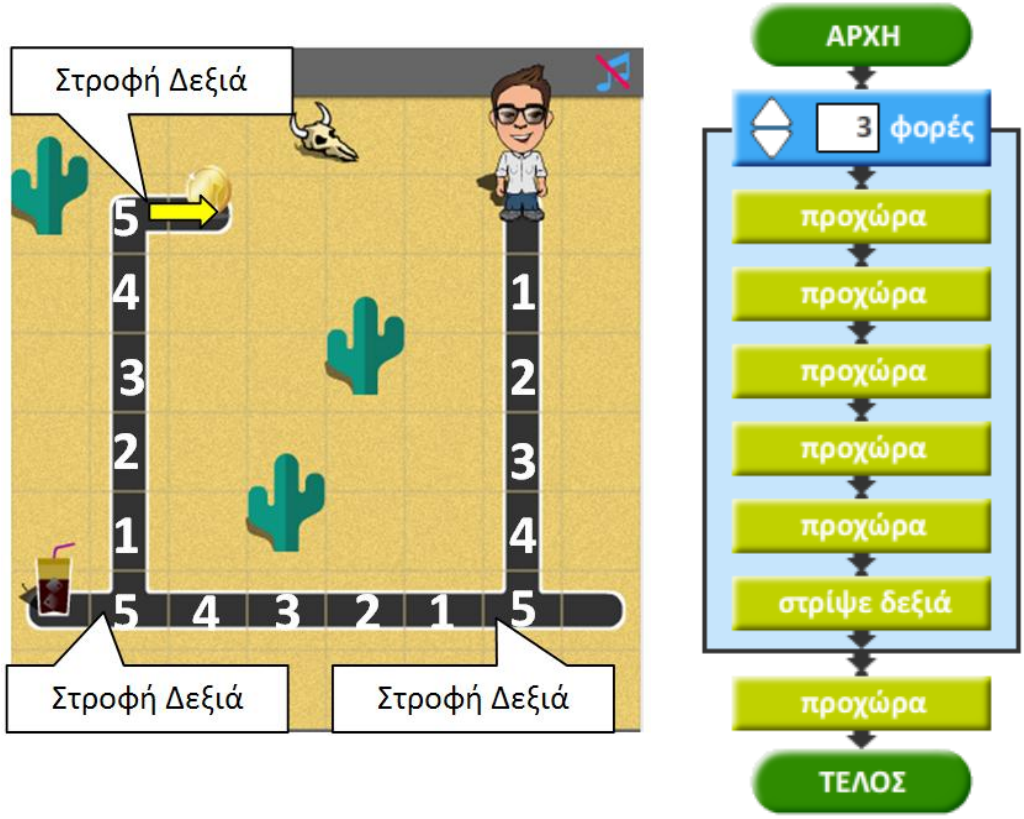
Σε αυτό το παράδειγμα οι δύο δομές πρέπει να επαναληφθούν 3 φορές η κάθε μια ώστε να φτάσουμε το Θαλή μέχρι το τελευταίο βήμα πριν το νόμισμα. Η πρώτη επανάληψη φτάνει τον Θαλή ακριβώς μέχρι τη μέση της διαδρομής.

Η δεύτερη επανάληψη φτάνει τον Θαλή ακριβώς ένα βήμα πριν το νόμισμα. Η τελευταία εντολή «προχώρα» οδηγεί τον Θαλή πάνω στο νόμισμα. Ανάλογα με το πρόβλημα που έχουμε να αντιμετωπίσουμε η δομή της επανάληψης χρησιμοποιείται διαφορετικά. Πολλές φορές μπορεί να μας μπερδέψει και ένα πρόβλημα να γίνει σπαζοκεφαλιά ενώ στην πραγματικότητα είναι πολύ απλό. Ένα παράδειγμα είναι η επόμενη πίστα του παιχνιδιού όπου θα πρέπει με μία εντολή επανάληψης μόνο να προγραμματίσουμε τον Θαλή ώστε να φτάσει στο νόμισμα (Σχ. 3.17).

Σχήμα 3.17: Εικόνα παιχνιδιού.

Η διαδρομή που θα πρέπει να ακολουθήσει ο Θαλής παρουσιάζεται απλή, όμως πως θα μπορούσαμε να προγραμματίσουμε το παιχνίδι ώστε να φτάσει στο νόμισμα με μία μόνο επανάληψη; Αν μελετήσουμε λίγο το μοτίβο της διαδρομής για αυτήν την πίστα του παιχνιδιού θα δούμε πως ο Θαλής θα πρέπει να

προχωρήσει 5 βήματα και έπειτα να στρίψει δεξιά. Αν το κάνει αυτό επαναλαμβανόμενα 3 φορές θα φτάσει ένα βήμα πριν το νόμισμα. Στην εικόνα του Σχήματος 3.18 έχουμε σχεδιάσει τη διαδρομή που πρέπει να κάνει ο Θαλής μέχρι το νόμισμα και έχουμε αριθμήσει τα βήματα. Ο κώδικας του προγράμματος παρουσιάζεται στην διπλανή εικόνα. Στο τέλος της επανάληψης χρησιμοποιούμε μία εντολή «προχώρα» ώστε να οδηγήσουμε τον Θαλή πάνω στο νόμισμα.



Σχήμα 3.18: Εικόνα παιχνιδιού και διάγραμμα ροής του κώδικα.

Πολλές φορές σε ένα πρόβλημα βρισκόμαστε αντιμέτωποι με μία ή περισσότερες διαφορετικές επιλογές που πρέπει να κάνουμε ώστε να το επιλύσουμε. Μέχρι στιγμής έχουμε γνωρίσει τη δομή της ακολουθίας και της επανάληψης και έχουμε αντιμετωπίσει προβλήματα που λύνονται με έναν τρόπο. Τι γίνεται όμως αν ένα πρόβλημα απαιτεί μία η περισσότερες επιλογές από εμάς ώστε να λυθεί; Για να μπορέσουμε να αντιμετωπίσουμε τέτοιες καταστάσεις στον προγραμματισμό υπάρχει η δομή της επιλογής.

3.4 Η δομή της επιλογής

Μπορούμε να φανταστούμε τη δομή της επιλογής σαν ένα μονοπάτι όπου διακλαδώνεται σε δύο διαφορετικά μονοπάτια και εμείς βρισκόμαστε στη μέση και πρέπει να διαλέξουμε ποιο από τα δύο μονοπάτια που βρίσκονται μπροστά μας θα ακολουθήσουμε. Η απόφαση που θα πάρουμε για το μονοπάτι που θα ακολουθήσουμε επηρεάζεται από διάφορα εμπόδια ή διάφορα γεγονότα που μπορεί να συμβαίνουν καθώς προχωράμε. Έτσι για παράδειγμα, αν βλέπουμε πως το ένα μονοπάτι δεν οδηγεί εκεί που εμείς θέλουμε να πάμε, προφανώς θα διαλέξουμε το άλλο. Αυτό που μας επηρεάζει ώστε να πάρουμε την απόφαση για το ποιο μονοπάτι θα διαλέξουμε, στον προγραμματισμό, καλείται **συνθήκη**. Η μορφή μίας δομής επιλογής στον προγραμματισμό είναι η ακόλουθη (Σχ. 3.19).

Σχήμα 3.19: Διάγραμμα ροής της δομής επιλογής.

Ανάλογα με τη συνθήκη που βρίσκεται στο κέντρο «π.χ. εμπόδιο ευθεία;» αν η συνθήκη αυτή ισχύει (ΝΑΙ), αν δηλαδή υπάρχει εμπόδιο μπροστά μας θα εκτελεστεί η εντολή «στρίψε αριστερά». Αν όμως η συνθήκη αυτή δεν ισχύει (ΟΧΙ) θα εκτελεστεί η εντολή «στρίψε δεξιά». Έπειτα από αυτή την διαδικασία, όποια και να είναι η επιλογή που έγινε θα εκτελεστεί η εντολή «προχώρα». Με αυτόν τον τρόπο μπορούμε να χρησιμοποιήσουμε τη δομή της επιλογής με διαφορετικές συνθήκες κάθε φορά ώστε να επιλύσουμε διάφορα προβλήματα. Ας δούμε ένα παράδειγμα χρήσης αυτής της δομής σε μία επόμενη πίστα του παιχνιδιού μας με το Θαλή και το νόμισμα.

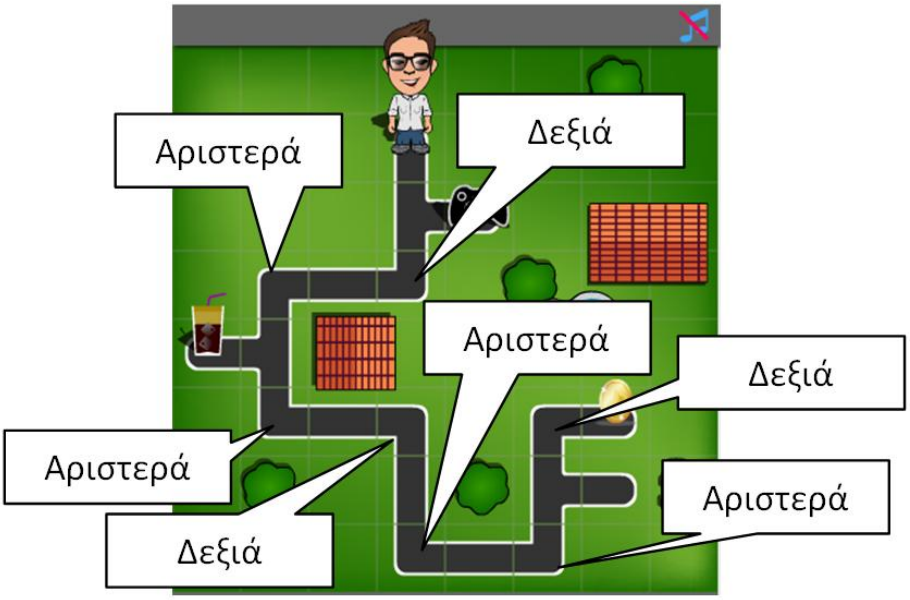
Σε αυτό το παράδειγμα θα πρέπει να οδηγήσουμε το Θαλή στο νόμισμα χρησιμοποιώντας οπωσδήποτε μία δομή επανάληψης. Εδώ δημιουργείται το ερώτημα πως θα γίνει αυτό με μία δομή επανάληψης αφού το μοτίβο της διαδρομής κάθε φορά αλλάζει και δεν είναι πάντοτε ίδιο όπως στα προηγούμενα

προβλήματα που αντιμετωπίσαμε; Τη λύση έρχεται να δώσει η δομή της επιλογής. Η δομή της επιλογής μας επιτρέπει να επιλέξουμε διάφορες συνθήκες και να κάνουμε διάφορες επιλογές. Στη συνέχεια απεικονίζονται οι συνθήκες που μπορούμε να επιλέξουμε (Σχ. 3.20).

Σχήμα 3.20: Εικόνα παιχνιδιού και επιλογές της συνθήκης.

Αν παρατηρήσουμε τη διαδρομή έως το νόμισμα θα δούμε πως ο Θαλής θα πρέπει πάντοτε να προχωρά δύο βήματα εμπρός και κάθε φορά να στρίβει αριστερά ή δεξιά ανάλογα σε ποιο τμήμα της διαδρομής βρίσκεται. Έτσι, στην πρώτη στροφή θα πρέπει να στρίψει δεξιά, στην δεύτερη αριστερά, στην τρίτη

πάλι αριστερά, στην τέταρτη δεξιά, στην πέμπτη αριστερά, στην έκτη αριστερά και στην έβδομη δεξιά όπως παρουσιάζεται στην ακόλουθη εικόνα (Σχ. 3.21).

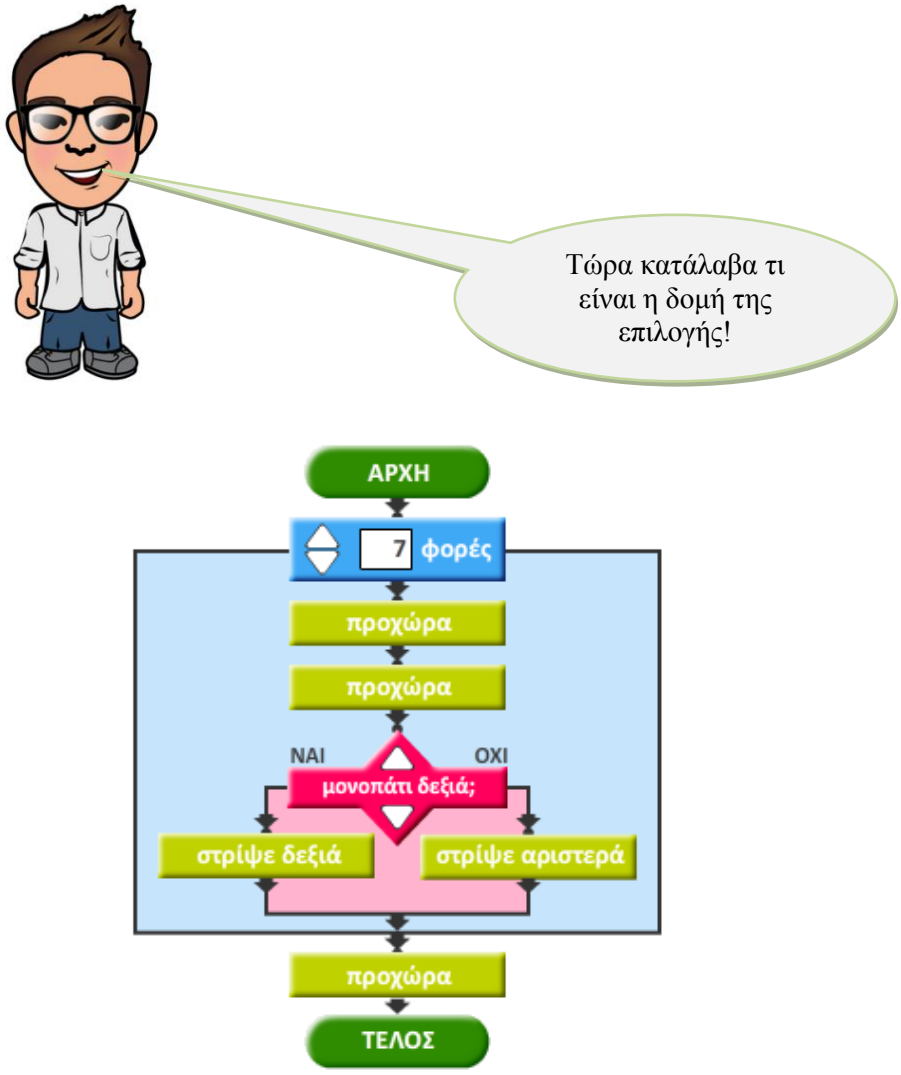


Σχήμα 3.21: Εικόνα παιχνιδιού.

Θα πρέπει λοιπόν μέσα στη δομή της επανάληψης να χρησιμοποιήσουμε μία δομή επιλογής που θα κάνει τον Θαλή να στρίβει δεξιά ή αριστερά όταν βρίσκεται σε κάποια από τις 7 στροφές της διαδρομής. Θα πρέπει λοιπόν να επιλέξουμε τη σωστή συνθήκη που θα μας δώσει λύση σε αυτό το πρόβλημα. Μία πολύ καλή επιλογή συνθήκης είναι η «**μονοπάτι δεξιά;**» όπου όταν ο Θαλής φτάνει σε μία στροφή θα ελέγχει αν υπάρχει μονοπάτι στα δεξιά του και αν υπάρχει θα στρίβει δεξιά αλλιώς θα στρίβει αριστερά (Σχ. 3.22).

Σχήμα 3.22: Επιλογές συνθήκης.

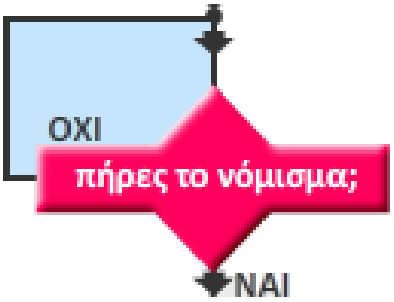
Επομένως ο Θαλής θα πρέπει να κάνει δύο βήματα εμπρός, να ελέγχει αν υπάρχει μονοπάτι δεξιά του, να στρίβει δεξιά ή αριστερά ανάλογα με τη συνθήκη και να επαναλαμβάνει την διαδικασία αυτή μέχρι να φτάσει ένα βήμα πριν το νόμισμα. Αφού βρίσκεται ένα βήμα πριν το νόμισμα με μία εντολή «προχώρα» θα μπορέσει να φτάσει σε αυτό. Πόσες φορές όμως πρέπει να επαναλάβουμε αυτή την διαδικασία; Η απάντηση για το συγκεκριμένο πρόβλημα είναι όσες στροφές υπάρχουν, δηλαδή επτά (7). Στη συνέχεια παρουσιάζεται ο κώδικας του προγράμματος αυτού (Σχ. 3.23).



Σχήμα 3.23: Διάγραμμα ροής του κώδικα.

3.5 Μία διαφορετική δομή επανάληψης

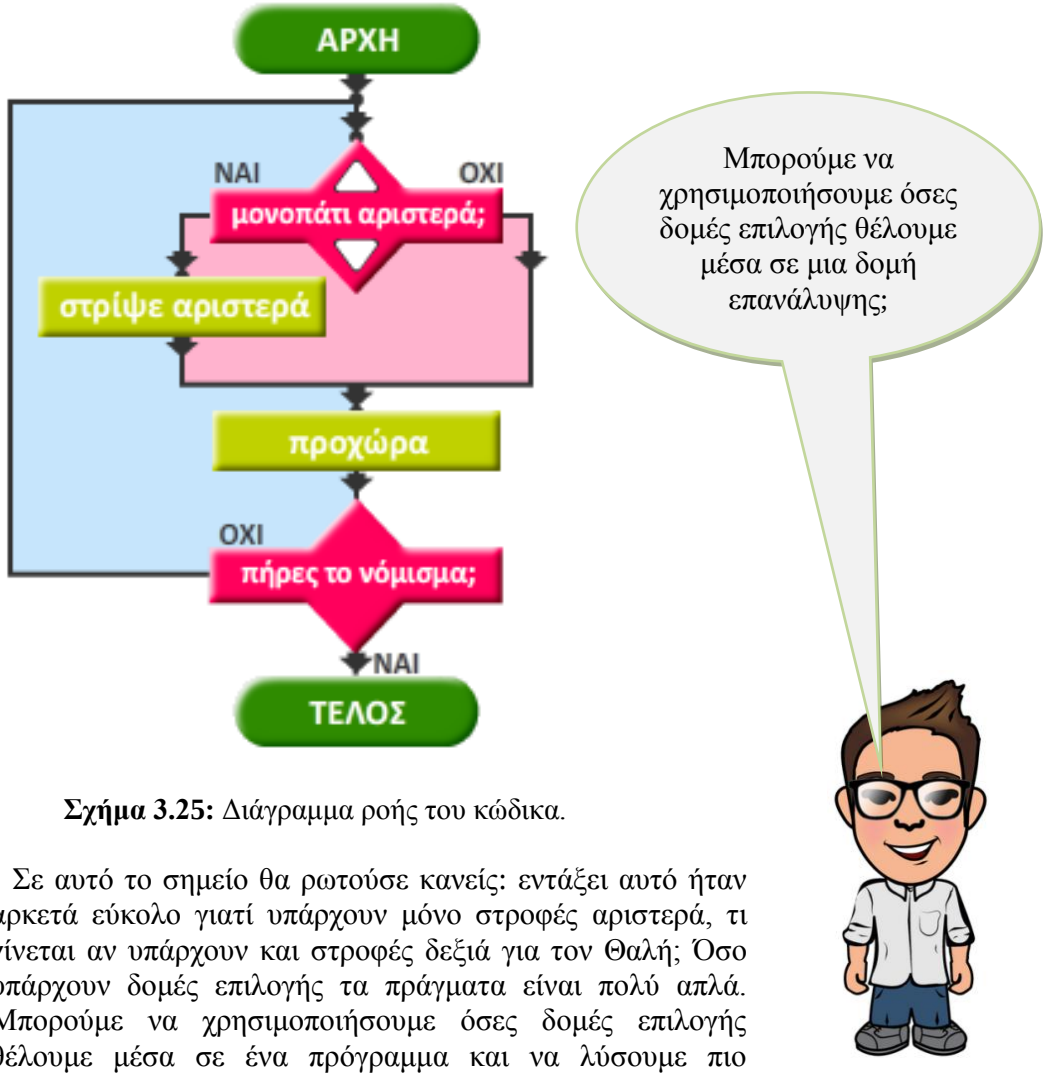
Πολλές φορές δεν γνωρίζουμε από την αρχή πόσες φορές επιθυμούμε να πραγματοποιήσουμε μία επανάληψη. Φανταστείτε λοιπόν στο παιχνίδι με τον Θαλή και το νόμισμα να πρέπει να εκτελέσετε διαρκώς μία διαδρομή μέχρι ο Θαλής να φτάσει και να πάρει το νόμισμα. Η δομή επανάληψης αυτή δεν προσδιορίζει τον αριθμό των επαναλήψεων αλλά ελέγχει αν ικανοποιείται μία συνθήκη ή όχι όπως «πήρες το νόμισμα;». Όσο ο Θαλής δεν έχει πάρει το νόμισμα η επανάληψη συνεχίζεται. Ας δούμε λοιπόν την επόμενη πίστα του παιχνιδιού μας όπου μας αναγκάζει να χρησιμοποιήσουμε την νέα επαναληπτική δομή (Σχ. 3.24).



Σχήμα 3.24: Εικόνα παιχνιδιού και διάγραμμα ροής του κώδικα.

Μελετώντας τη διαδρομή που πρέπει να ακολουθήσει ο Θαλής θα δούμε πως έχει μόνο στροφές αριστερά. Επομένως αν δίναμε εντολή στον Θαλή να ελέγχει

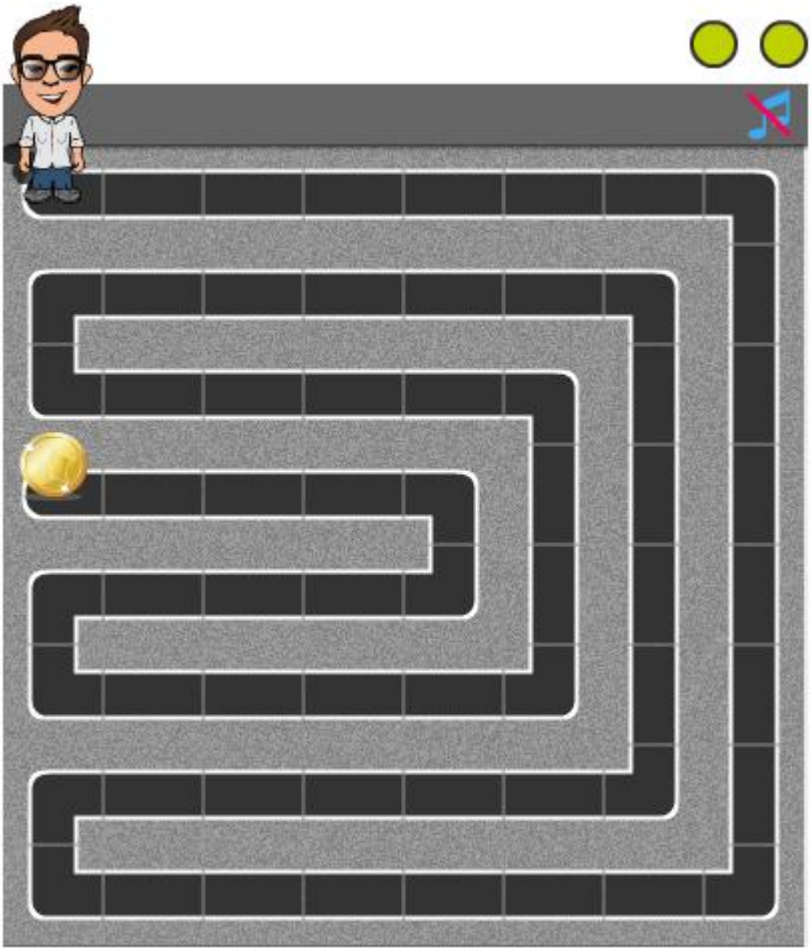
αν υπάρχει μονοπάτι αριστερά του να στρίβει αριστερά αλλιώς να προχωρά ένα βήμα τότε θα έφτανε στο νόμισμα και η επανάληψη θα τερματιζόταν όταν αυτός έπαιρνε το νόμισμα. Αν δεν υπάρχει στροφή αριστερά, η δομή επιλογής δεν εκτελεί κάποια ενέργεια και συνεχίζει με την επόμενη εντολή «προχώρα» οδηγώντας τον Θαλή ένα βήμα ποιο κοντά στο νόμισμα. Αν υπάρχει μονοπάτι αριστερά του Θαλή τότε εκτελείται η εντολή «στρίψε αριστερά» και έπειτα η εντολή «προχώρα». Η διαδικασία επαναλαμβάνεται έως ότου ο Θαλής πάρει το νόμισμα (Σχ. 3.25).



Σχήμα 3.25: Διάγραμμα ροής του κώδικα.

Σε αυτό το σημείο θα ρωτούσε κανείς: εντάξει αυτό ήταν αρκετά εύκολο γιατί υπάρχουν μόνο στροφές αριστερά, τι γίνεται αν υπάρχουν και στροφές δεξιά για τον Θαλή; Όσο υπάρχουν δομές επιλογής τα πράγματα είναι πολύ απλά. Μπορούμε να χρησιμοποιήσουμε όσες δομές επιλογής θέλουμε μέσα σε ένα πρόγραμμα και να λύσουμε πιο σύνθετα προβλήματα κάνοντας συνδυασμό των τριών δομών: της ακολουθίας, της επιλογής και της επανάληψης.

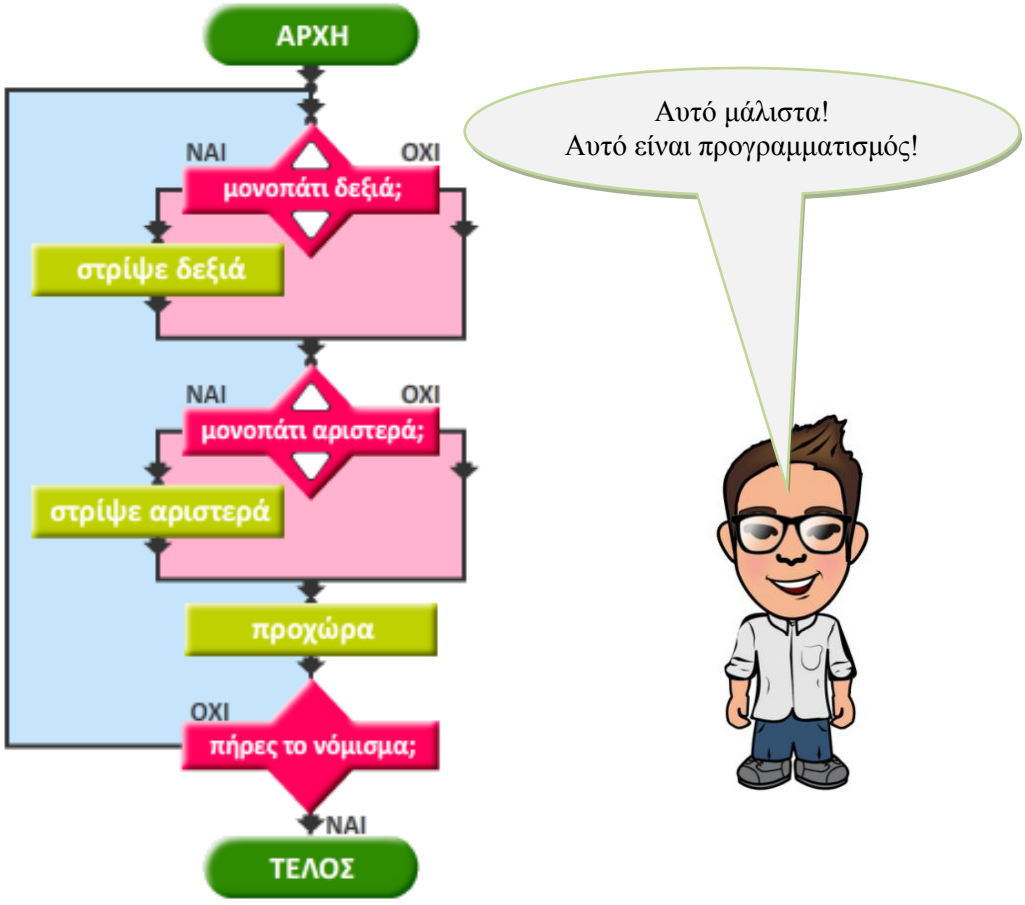
Ας δούμε ένα παράδειγμα από την επόμενη πίστα του παιχνιδιού μας (Σχ. 3.26).



Σχήμα 3.26: Εικόνα παιχνιδιού.

Εδώ τα πράγματα μοιάζουν με λαβύρινθο. Στην πραγματικότητα όσο δύσκολο και να παρουσιάζεται είναι πολύ απλό. Είναι η ίδια διαδικασία με την προηγούμενη πίστα του παιχνιδιού μόνο που πρέπει να υπολογίσουμε πως υπάρχουν και στροφές δεξιά και αριστερά. Επομένως αν χρησιμοποιήσουμε άλλη μία δομή επιλογής στο προηγούμενο πρόγραμμα όπου θα ελέγχει και για τις δεξιές στροφές το πρόβλημά μας θα λυθεί άμεσα και θα κάνουμε τον Θαλή να λύσει τον λαβύρινθο και να πάρει τον θησαυρό. Η πρώτη δομή επιλογής ελέγχει αν υπάρχει μονοπάτι δεξιά και οδηγεί τον Θαλή να στρίψει δεξιά ενώ η δεύτερη δομή επιλογής ελέγχει αν υπάρχει μονοπάτι στα αριστερά και οδηγεί τον Θαλή να

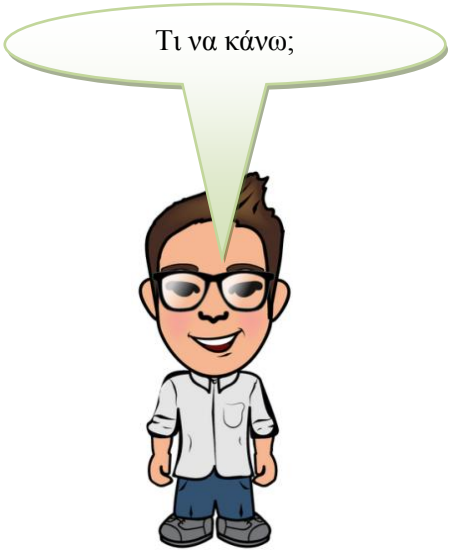
στρίψει αριστερά. Έπειτα εκτελεί μία εντολή «προχώρα». Η διαδικασία αυτή επαναλαμβάνεται εωσότου ο Θαλής φτάσει το νόμισμα (Σχ. 3.27).



Σχήμα 3.27: Διάγραμμα ροής του κώδικα.

Υπάρχουν προβλήματα που απαιτούν περισσότερες επιλογές για να επιλυθούν και δυσκολεύουν περισσότερο τον προγραμματισμό τους. Σε αυτά τα προβλήματα πρέπει να χρησιμοποιούμε δομές επιλογής με διάφορες συνθήκες. Επιλέγοντας κατάλληλη συνθήκη επιλογής το πρόγραμμά μας μπορεί να λυθεί πολύ εύκολα, ενώ επιλέγοντας μία λιγότερη κατάλληλη συνθήκη μπορεί να δυσκολευτούμε να προγραμματίσουμε τη λύση του προβλήματος μας. Ας δούμε ένα παράδειγμα της τελευταίας πίστας του παιχνιδιού μας με τον Θαλή (Σχ. 3.28).

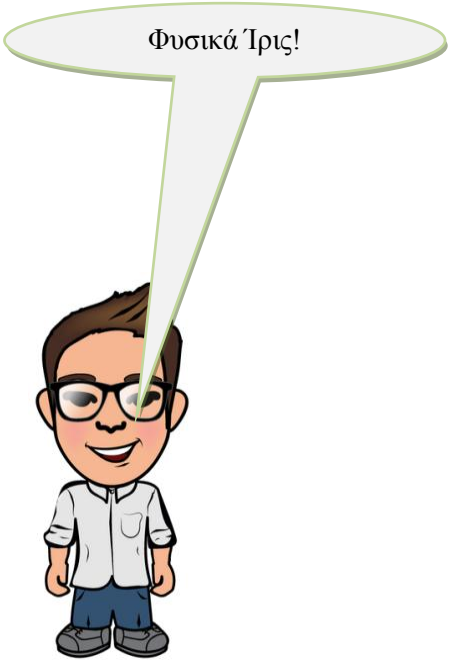
Σχήμα 3.28: Εικόνα παιχνιδιού.

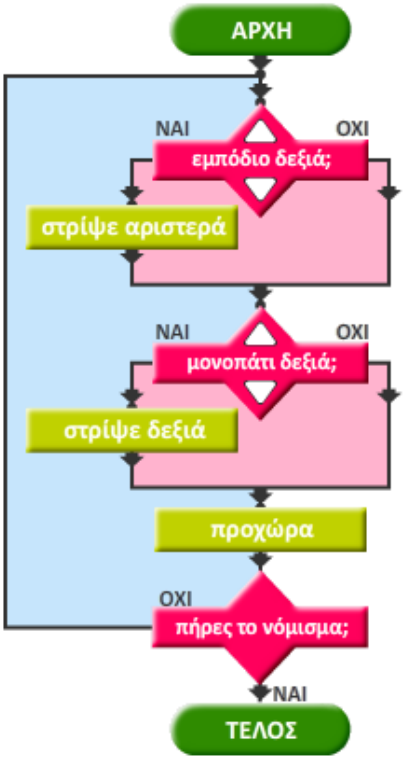


Σε αυτή την πίστα ο Θαλής έχει να αντιμετωπίσει εκτός από την σωστή διαδρομή και πολλά εμπόδια στον δρόμο του για το νόμισμα. Εφόσον υπάρχουν πολλά εμπόδια καθώς και διάφορα μονοπάτια αν μελετήσουμε λίγο την διαδρομή θα παρατηρήσουμε πως μπορούμε να χρησιμοποιήσουμε τη δομή επιλογής για να αποφύγουμε εμπόδια στο δρόμο μας και τη δομή επιλογής για να ακολουθήσουμε μονοπάτια δίνοντας εντολή κάθε φορά στο Θαλή να στρίψει προς την κατάλληλη κατεύθυνση.

Μία λύση του προβλήματος είναι να χρησιμοποιήσουμε μέσα στη δομή της επανάληψης δύο δομές επιλογής που ελέγχουν αν υπάρχει εμπόδιο στα δεξιά του Θαλή ή αν υπάρχει μονοπάτι στα

δεξιά του Θαλή. Αν υπάρχει εμπόδιο στα δεξιά του τότε δίνουμε εντολή στον Θαλή να στρίψει αριστερά για να μην πέσει πάνω στο εμπόδιο. Αν υπάρχει μονοπάτι στα δεξιά του Θαλή τότε λέμε στον Θαλή να στρίψει δεξιά για να ακολουθήσει αυτό το μονοπάτι. Μελετώντας τη διαδρομή στην εικόνα θα δούμε ότι στα δεξιά του Θαλή θα υπάρχει πάντα ένα μονοπάτι ή ένα εμπόδιο μέχρι το νόμισμα. Επομένως μία λύση του προβλήματος είναι αυτή που παρουσιάζεται στον κώδικα που γράψαμε. Πάντα μετά τον έλεγχο των συνθηκών ο Θαλής προχωρά ένα βήμα και η διαδικασία επαναλαμβάνεται έως ότου πάρει το νόμισμα (Σχ. 3.29).





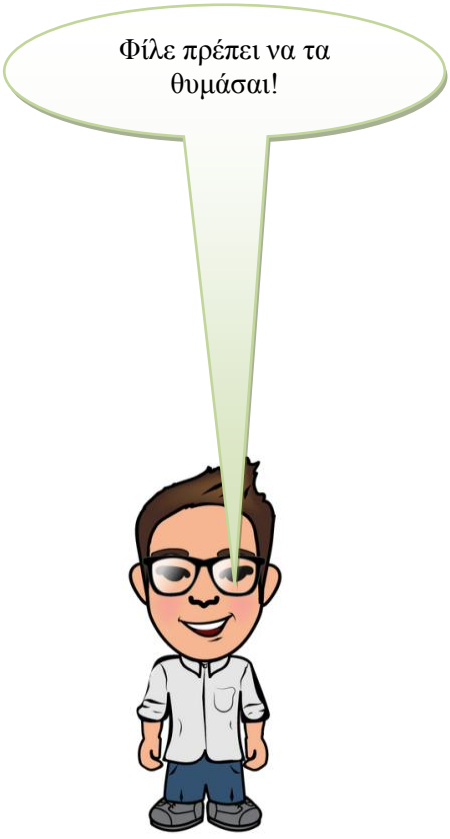
Σχήμα 3.29: Διάγραμμα ροής του κώδικα.

3.6 Πίσω από τα παιχνίδια και τα διαγράμματα

Μέχρι τώρα έχουμε μελετήσει τον προγραμματισμό ενός παιχνιδιού με τον Θαλή και το νόμισμα και έχουμε χρησιμοποιήσει διαγράμματα ροής για να δώσουμε λύση στα προβλήματα που αντιμετωπίζαμε καθώς προχωρούσαμε στις πίστες του παιχνιδιού. Τα διαγράμματα ροής μας βοηθούν να καταλάβουμε με γραφικό τρόπο τη λειτουργία ενός προγράμματος.

Τα προβλήματα που αντιμετωπίσαμε μέχρι τώρα ήταν απλά προβλήματα εύρεσης της σωστής διαδρομής όπου χρησιμοποιήσαμε τις βασικές δομές (ακολουθία, επιλογή και επανάληψη) και μερικές βασικές εντολές για να οδηγήσουμε τον Θαλή στο νόμισμα. Πίσω όμως από τα διαγράμματα κρύβεται **ο κώδικας** του προγράμματος ο οποίος στην πραγματικότητα έχει μεταμορφωθεί σε διάγραμμα για να είναι ευκολότερο να κατανοηθεί. Από όσα είπαμε δημιουργούνται τα ακόλουθα ερωτήματα:

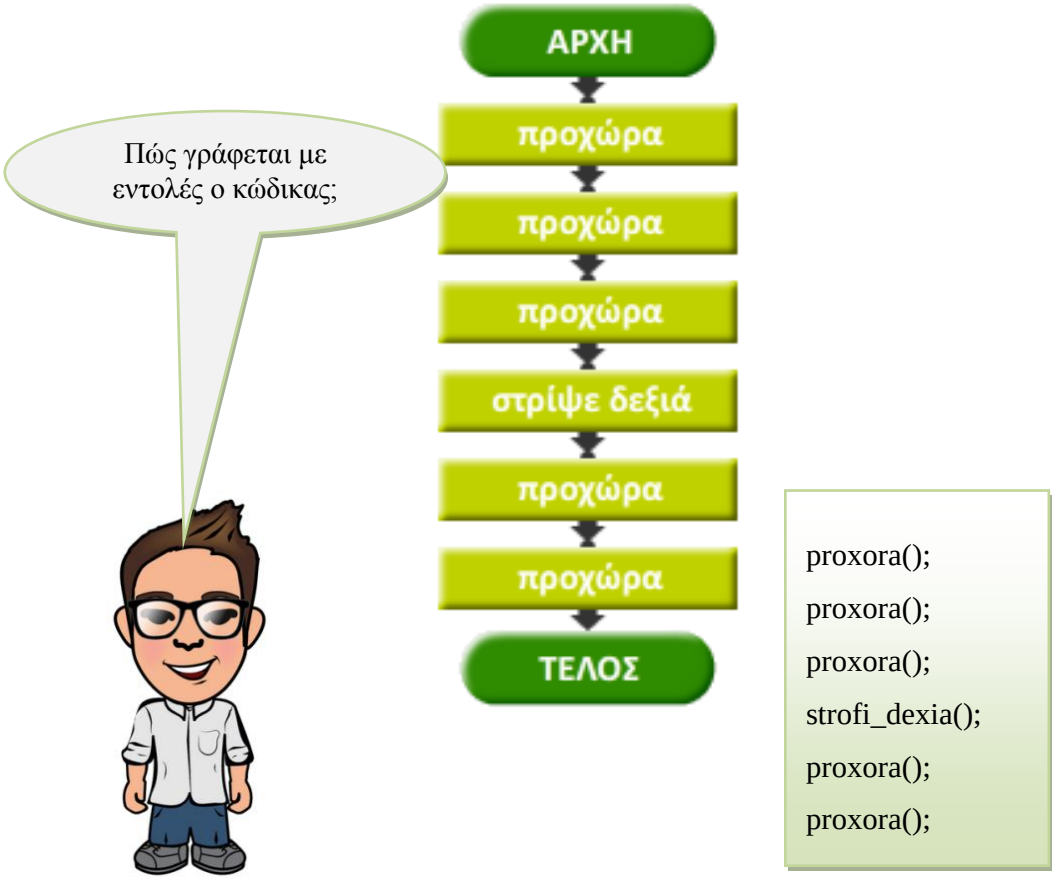
Τι είναι ο κώδικας;
Ο κώδικας είναι ένα σύνολο εντολών που έχουν γραφεί σύμφωνα με τους κανόνες που ορίζει το συντακτικό μίας γλώσσας προγραμματισμού.



Τι είναι η γλώσσα προγραμματισμού;
Μία γλώσσα προγραμματισμού ορίζει ένα σύνολο εντολών καθώς και ένα συντακτικό για αυτές (τις εντολές) και τη χρησιμοποιούμε για να γράφουμε προγράμματα στους ηλεκτρονικούς υπολογιστές.

Τί είναι το συντακτικό μίας γλώσσας προγραμματισμού;
Όπως και στην γραμματική της Ελληνικής γλώσσας ή της Αγγλικής γλώσσας υπάρχει συντακτικό που ορίζει πώς θα γράφονται οι λέξεις και οι προτάσεις μέσα σε μία έκθεση ή κάποιο κείμενο, έτσι και στην γλώσσα προγραμματισμού υπάρχει ένα συντακτικό που ορίζει πώς θα γράφονται οι εντολές ενός προγράμματος.

Καλά ως εδώ αλλά ας δούμε ένα πραγματικό παράδειγμα κώδικα και πως αυτός λειτουργεί ώστε να εκτελεί εντολές ενός προγράμματος. Ας πάρουμε το πρώτο παράδειγμα κώδικα με διάγραμμα που γράψαμε στην πρώτη πίστα του παιχνιδιού μας. Σε αυτό το παράδειγμα έπρεπε να εκτελέσουμε 6 εντολές για να οδηγήσουμε τον Θαλή στο νόμισμα. Στη συνέχεια παρουσιάζεται η σύνταξη αυτού του προγράμματος με κώδικα σε μορφή κειμένου που εκτελεί την ίδια διαδικασία με αυτή του διαγράμματος (Σχ. 3.30).



Σχήμα 3.30: Διάγραμμα ροής και κώδικας.

Πολύ εύκολα μπορούμε να καταλάβουμε πως το παράδειγμα σε μορφή κώδικα με κείμενο μοιάζει πολύ με τον κώδικα του διαγράμματος. Στην ουσία οι εντολές που εκτελούμε έχουν γραφεί με κείμενο όπου χρησιμοποιούνται μόνο λατινικοί χαρακτήρες και υπάρχει μία ιδιαίτερη σύνταξη με παρενθέσεις και ερωτηματικά. Εδώ δημιουργούνται οι ακόλουθες ερωτήσεις.

Το ερωτηματικό στον κώδικα σημαίνει ερώτηση;

Όχι, το συντακτικό της γλώσσας προγραμματισμού ορίζει πως το Ελληνικό ερωτηματικό (;) σημαίνει το τέλος μίας εντολής και πρέπει να μπαίνει πάντοτε ώστε να ξεχωρίζουμε πότε τελειώνει μία εντολή μέσα στον κώδικα.

Γιατί στο τέλος των εντολών υπάρχει πάντα μία παρένθεση;

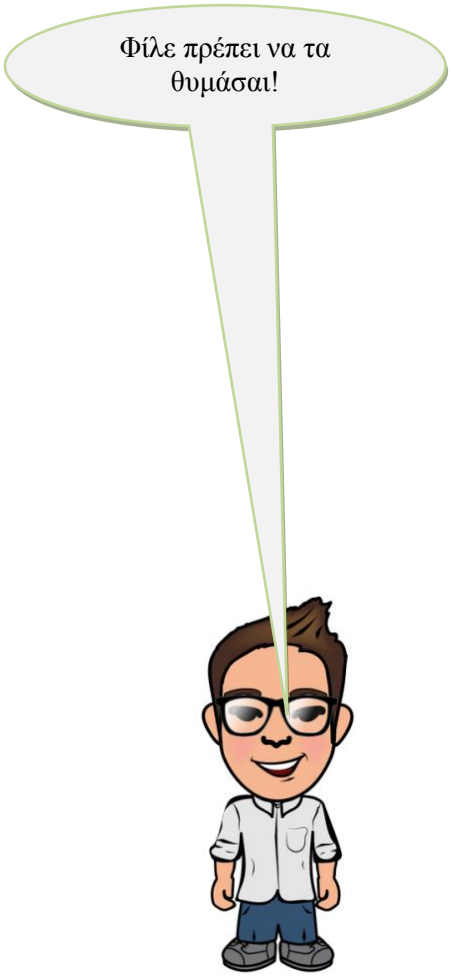
Οι παρενθέσεις δεν υπάρχουν πάντα. Οι παρενθέσεις σημαίνουν πως αυτό που εκτελείται είναι μία διαδικασία και όχι μία μόνο εντολή.

Τι είναι διαδικασία;

Μία διαδικασία είναι ένα σύνολο από εντολές που εκτελούνται και δίνουν λύση σε ένα πρόβλημα. Στο παράδειγμά μας με το παιχνίδι η διαδικασία proxora(); εκτελεί ένα σύνολο εντολών που κάνει τον Θαλή να προχωρά ένα βήμα μπροστά μέσα στην πίστα του παιχνιδιού (ένα τετράγωνο).

Ποιες εντολές κρύβει μέσα της μία διαδικασία; Γιατί δεν τις βλέπουμε;

Οι εντολές που κρύβονται μέσα σε μία διαδικασία δεν φαίνονται σε εμάς ακόμα γιατί θα πρέπει να έχουμε περισσότερη εμπειρία στον προγραμματισμό για να τις καταλάβουμε. Επομένως, για τις διαδικασίες που χρειάζονται πολλές εντολές για να εκτελεστούν, όπως για παράδειγμα η διαδικασία όπου ο Θαλής προχωρά ένα βήμα μπροστά ή στρίβει δεξιά και αριστερά, έχουν γραφεί από



έμπειρους προγραμματιστές και τους έχουν αποδοθεί ονόματα όπως proxora(), strofi_dexia() και strofi_aristera() για να μας διευκολύνουν στον προγραμματισμό του παιχνιδιού. Εμείς πλέον χρησιμοποιώντας αυτές τις διαδικασίες με σωστή σειρά και σωστή σύνταξη μπορούμε να κινούμε τον Θαλή μέσα στο παιχνίδι και να τον οδηγούμε στο νόμισμα.

3.7 Γράφοντας μεγαλύτερα προγράμματα με κώδικα

Ας δούμε τώρα ένα μεγαλύτερο πρόγραμμα σε μορφή κώδικα που είναι αυτό της δεύτερης πίστας του παιχνιδιού μας (Σχ. 3.31). Σε αυτό το πρόγραμμα έχουμε 10 εντολές να εκτελέσουμε και όπως και στο προηγούμενο παράδειγμα το μόνο που έχουμε να γράψουμε με κώδικα είναι όλες οι εντολές ακολουθιακά χρησιμοποιώντας τις διαδικασίες poxora(), strofi_aristera() και strofi_dexia() που γνωρίσαμε προηγουμένως.

```
proxora();
strofi_aristera();
proxora();
proxora();
strofi_dexia();
proxora();
proxora();
strofi_aristera();
proxora();
proxora();
```

Σχήμα 3.31: Κώδικας και διάγραμμα ροής.

3.8 Συντακτικά λάθη

Πολλοί θα ρωτούσαν σε αυτό το σημείο, τι γίνεται αν γράψω τις διαδικασίες και τις εντολές με τυπογραφικά λάθη; Όπως είπαμε το συντακτικό μίας γλώσσας προγραμματισμού ορίζει το πώς θα γράφονται οι εντολές και οι διαδικασίες μέσα σε ένα πρόγραμμα. Το συντακτικό της γλώσσας προγραμματισμού που χρησιμοποιούμε για να προγραμματίσουμε το παιχνίδι με κώδικα, έχει τους ακόλουθους συντακτικούς κανόνες:

1. Όλες οι εντολές και οι διαδικασίες πρέπει να γράφονται με λατινικούς χαρακτήρες.
2. Δεν μπορούμε να χρησιμοποιούμε στην σύνταξη των διαδικασιών τους χαρακτήρες που φαίνονται στον ακόλουθο πίνακα καθώς και χαρακτήρες που έχουν συγκεκριμένη σημασία στον κώδικα όπως το ‘;’.

%	**$**	**#**	**!**	**^**	**&**	*****	**-**	**+**	**=**	**/**

3. Πρέπει πάντοτε στο τέλος των διαδικασιών να χρησιμοποιούμε τις παρενθέσεις () και στο τέλος κάθε εντολής ή διαδικασίας το Ελληνικό ερωτηματικό ‘;’.

Οι συντακτικοί κανόνες μίας γλώσσας προγραμματισμού είναι πάρα πολλοί και όταν γράφουμε κώδικα θα πρέπει να τους τηρούμε ώστε να μην δημιουργούνται προβλήματα στην εκτέλεση των προγραμμάτων. Στον Πίνακα 3.1 παρουσιάζουμε παραδείγματα συντακτικών σφαλμάτων στον κώδικα ενός προγράμματος.

Πίνακας 3.1: Λάθη σύνταξης.

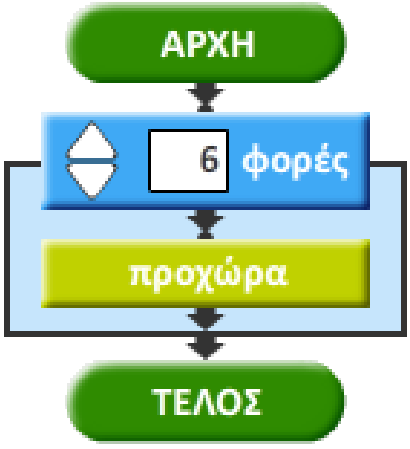
Σωστή σύνταξη	Λάθος Σύνταξη	Περιγραφή
proxora();	proxora()	Δεν έχουμε βάλει το ερωτηματικό ‘;’ στο τέλος της διαδικασίας.
strofi_dexia();	strofi-dexia();	Έχουμε βάλει τον μη επιτρεπτό χαρακτήρα ‘-‘ στην εντολή
strofi_aristera();	strofi_aristera(;	Δεν έχουμε κλείσει την παρένθεση στην διαδικασία όπως ορίζει το συντακτικό.

Όσο προχωράμε θα μαθαίνουμε περισσότερα για τα συντακτικά λάθη σε μία γλώσσα προγραμματισμού και θα γινόμαστε ποιο έμπειροι.

3.9 Δομές επανάληψης και επιλογής με κώδικα

Τώρα που καταλάβαμε βασικά ζητήματα γραφής κώδικα και πως αυτός μπορεί να συνταχθεί για να γράψουμε ένα πρόγραμμα είμαστε έτοιμοι να προχωρήσουμε ένα βήμα πιο βαθιά στον προγραμματισμό με κώδικα και να δούμε πως συντάσσονται οι δομές επανάληψης και επιλογής.

Η πρώτη δομή επανάληψης που συναντήσαμε στο παιχνίδι μας ήταν στην τρίτη πίστα όπου έπρεπε να οδηγήσουμε τον Θαλή στο νόμισμα επαναλαμβάνοντας την εντολή «προχώρα» 6 φορές. Ας δούμε όμως πως συντάσσεται αυτή η δομή επανάληψης με κώδικα (Σχ. 3.32).




```
for (var fores_1 = 0;   fores_1 < 6;   fores_1++)
{
  proxora();
}
```

Σχήμα 3.32: Διάγραμμα ροής και κώδικας σε γλώσσα C.

Παρατηρώντας για πρώτη φορά αυτή τη σύνταξη μπορεί να χάσουμε λίγο το κουράγιο μας στο να μάθουμε προγραμματισμό, όμως στην πραγματικότητα τα πράγματα δεν είναι και τόσο δύσκολα να τα κατανοήσουμε. Ο κώδικας που βλέπουμε πιο πάνω θα μπορούσε να μεταφραστεί όπως το ακόλουθο κείμενο:

Δήλωσε μία μεταβλητή με όνομα **fores_1** και ξεκινώντας από την τιμή **0** επανέλαβε τη διαδικασία **proxora()** αυξάνοντας την τιμή της μεταβλητής **fores_1** κατά **1** σε κάθε επανάληψη μέχρι η τιμή της μεταβλητής **fores_1** να γίνει μεγαλύτερη από **6.**

Ας αναλύσουμε λίγο καλύτερα τον κώδικα που μόλις περιγράψαμε.

for → Η Αγγλική λέξη for που σημαίνει 'για' και χρησιμοποιείται ως δεσμευμένη λέξη για αυτή τη δομή επανάληψης στις περισσότερες γλώσσες προγραμματισμού.

var → Είναι μία δεσμευμένη λέξη που βγαίνει από την Αγγλική λέξη variable που σημαίνει 'μεταβλητή'. Χρησιμοποιείται στον προγραμματισμό για να δηλώσει μία μεταβλητή η οποία μπορεί να αποθηκεύει διάφορες τιμές. Η μεταβλητή θα πρέπει να έχει ένα όνομα το οποίο δεν ανήκει σε κάποια δεσμευμένη λέξη της γλώσσας προγραμματισμού και θα πρέπει να ακολουθεί τους συντακτικούς κανόνες που ορίζει η γλώσσα προγραμματισμού. Σε αυτό το παράδειγμα που μελετάμε η μεταβλητή έχει όνομα **fores_1**.

++ → Όταν στον κώδικα συναντάμε τα δύο θετικά πρόσημα (++) δίπλα από μία μεταβλητή σημαίνει πως η μεταβλητή θα πρέπει να αυξήσει την τιμή της κατά 1 όταν έρθει η ώρα να εκτελεστεί αυτή η εντολή. Σε αυτό το παράδειγμα αυξάνουμε κατά 1 την τιμή της μεταβλητής **fores_1** χρησιμοποιώντας την εντολή **fores_1++.**

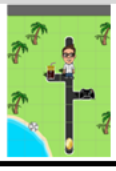
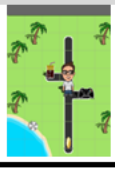
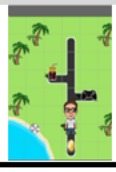
Η ακόλουθη γραμμή κώδικα μεταφράζεται όπως παρουσιάζεται στη συνέχεια:

```
for (var fores_1 = 0;   fores_1 < 6;   fores_1++)
```

Για (μεταβλητή fores_1 από 0 έως fores_1 γίνει μεγαλύτερο από 6 επανέλαβε και σε κάθε επανάληψη αύξησε την τιμή της μεταβλητής fores_1 κατά 1).

Η δομή της επανάληψης for επαναλαμβάνει ότι βρίσκεται μέσα στις αγκύλες {} όπου στο συγκεκριμένο παράδειγμα είναι η διαδικασία proxora();

Ας δούμε βήμα - βήμα τη διαδικασία επανάληψης και ας καταγράψουμε τα βήματα που επαναλαμβάνονται καθώς και όλες τις τιμές που παίρνει η μεταβλητή fores_1 (Σχ. 3.33).

Επανάληψη διαδικασίας proxora();							
Επανάληψη	1η	2η	3η	4η	5η	6η	7η
Τιμή fores_1	0	1	2	3	4	5	6
fores_1 < 6 ;	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΝΑΙ	ΟΧΙ
Αποτέλεσμα							

Σχήμα 3.33: Σχηματική παράσταση διαδικασία επανάληψης.

Μελετώντας τον πιο πάνω σχήμα παρατηρούμε πως καθώς εκτελείται η δομή της επανάληψης ελέγχεται αν η τιμή της μεταβλητής fores_1 είναι μικρότερη του αριθμού 6 ξεκινώντας την απαρίθμηση από την τιμή 0. Στην 7η επανάληψη γίνεται έλεγχος της συνθήκης **fores_1 < 6** (6 < 6) και εφόσον η συνθήκη δεν ισχύει (ΟΧΙ) η επανάληψη σταματά να εκτελείται και ο Θαλής έχει φτάσει το νόμισμα.

Η επανάληψη δεν ισχύει μόνο για μία διαδικασία. Μπορούμε να επαναλαμβάνουμε πολλές διαδικασίες και εντολές ταυτόχρονα με μία δομή επανάληψης

αρκεί να τοποθετήσουμε ότι θέλουμε να επαναλάβουμε μέσα στις αγκύλες {} της δομής επανάληψης. Ας δούμε ένα παράδειγμα επανάληψης που αντιστοιχεί στην 4η πίστα του παιχνιδιού μας. Σε αυτό το παράδειγμα πρέπει να επαναλάβουμε ένα σύνολο κινήσεων χρησιμοποιώντας τη δομή επανάληψης (Σχ. 3.34).


```
for (var fores_1 = 0;   fores_1 < 6;   fores_1++)
{
   proxora();
   strofi_aristera();
   proxora();
   strofi_dexia();
}
proxora();
```

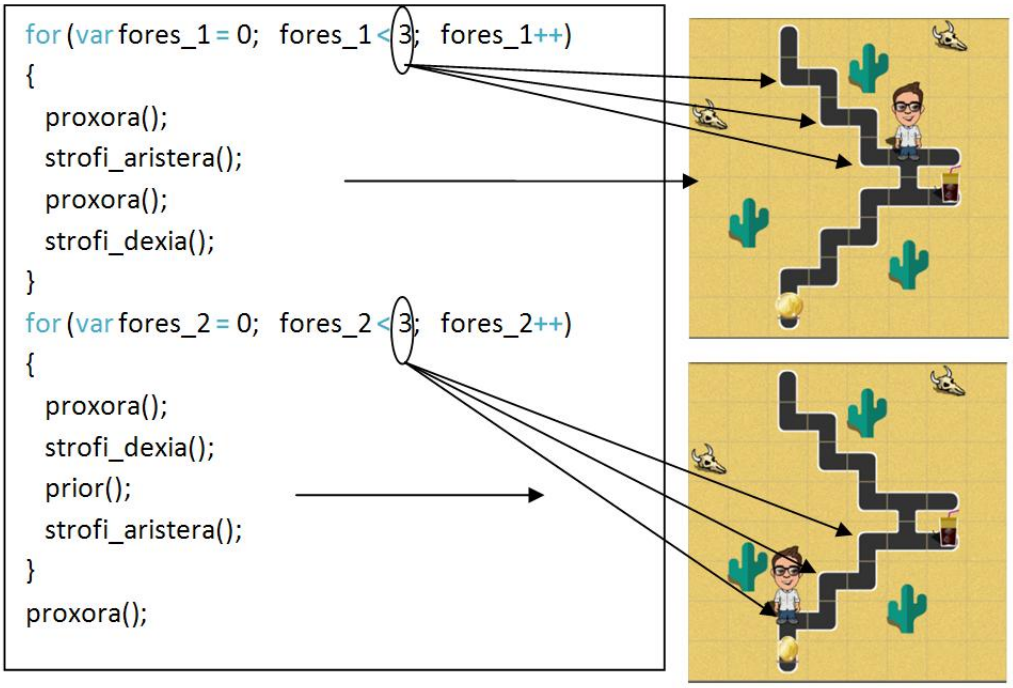
Σχήμα 3.34: Διάγραμμα ροής και κώδικας σε γλώσσα C.

Η επανάληψη σε αυτό το παράδειγμα θα πρέπει να εκτελεστεί επίσης 6 φορές και γι' αυτό η σύνταξη της δομής στις παρενθέσεις είναι ακριβώς η ίδια με το προηγούμενο παράδειγμα. Και πάλι χρησιμοποιούμε τη μεταβλητή

fores_1 για να απαριθμούμε και να ελέγχουμε τις επαναλήψεις που εκτελούνται κάθε φορά. Μέσα στις αγκύλες έχουμε τοποθετήσει το μοτίβο κινήσεων που θέλουμε να επαναλαμβάνει ο Θαλής ώστε να ακολουθήσει τη σωστή διαδρομή προς το νόμισμα. Η επανάληψη ισχύει για τις διαδικασίες που βρίσκονται μέσα στις αγκύλες και μόνο. Μετά το τέλος της επανάληψης, έξω από τις αγκύλες υπάρχει μία διαδικασία proxora(); η οποία θα εκτελεστεί ώστε να γίνει το τελευταίο βήμα του Θαλή προς το νόμισμα. Θα πρέπει να προσέχουμε λοιπόν τα σύνορα της δομής επανάληψης μέσα σε ένα πρόγραμμα τα οποία ορίζονται από τις αγκύλες ώστε να μην επαναλαμβάνουμε διαδικασίες που δεν χρειάζεται να επαναληφθούν (Σχ. 3.35).

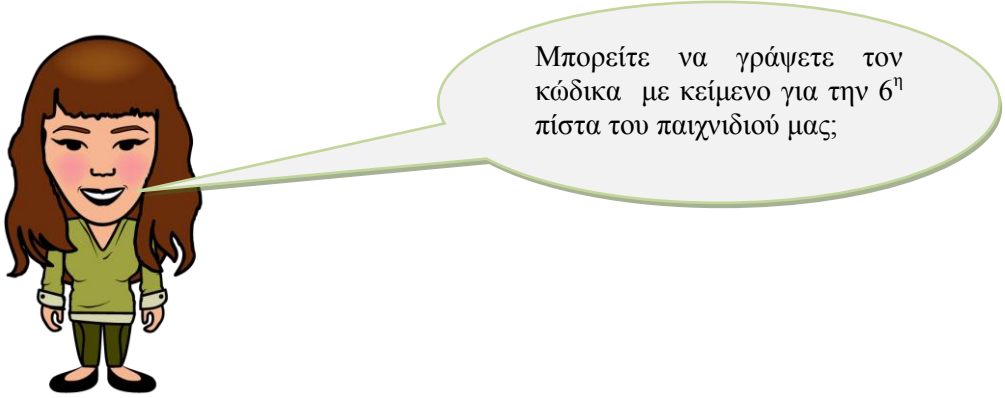
Σε ένα πρόγραμμα όμως μπορούμε να χρησιμοποιήσουμε περισσότερες από μία δομές επανάληψης αν αυτό είναι απαραίτητο όπως στο παράδειγμα της 5^{ης} πίστας του παιχνιδιού μας όπου έπρεπε να επαναλάβουμε δύο μοτίβα κινήσεων για να οδηγήσουμε τον Θαλή στο νόμισμα. Σε αυτό το παράδειγμα χρησιμοποιούμε δύο επαναλήψεις for μέσα στον κώδικά μας και δηλώνουμε δύο μεταβλητές fores_1 και fores_2 για να διαχειριστούμε τις δύο επαναλήψεις ξεχωριστά. Στη συνέχεια παρουσιάζουμε τον κώδικα του παραδείγματος αυτού (Σχ. 3.36).

Σχήμα 3.35: Διάγραμμα ροής του κώδικα.

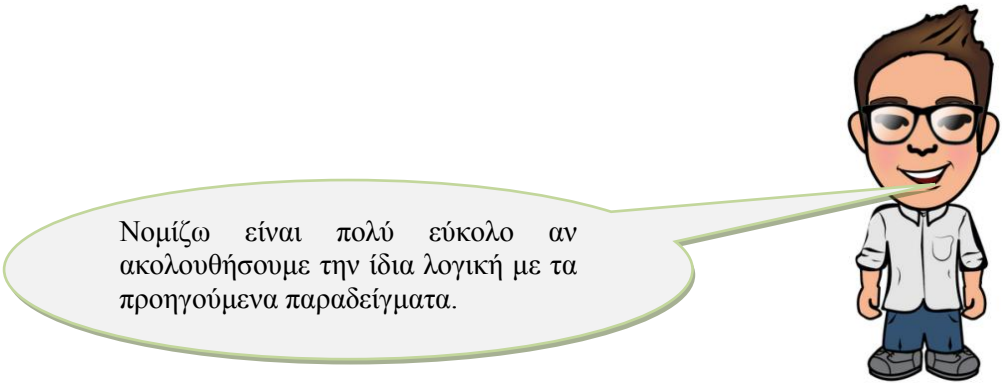


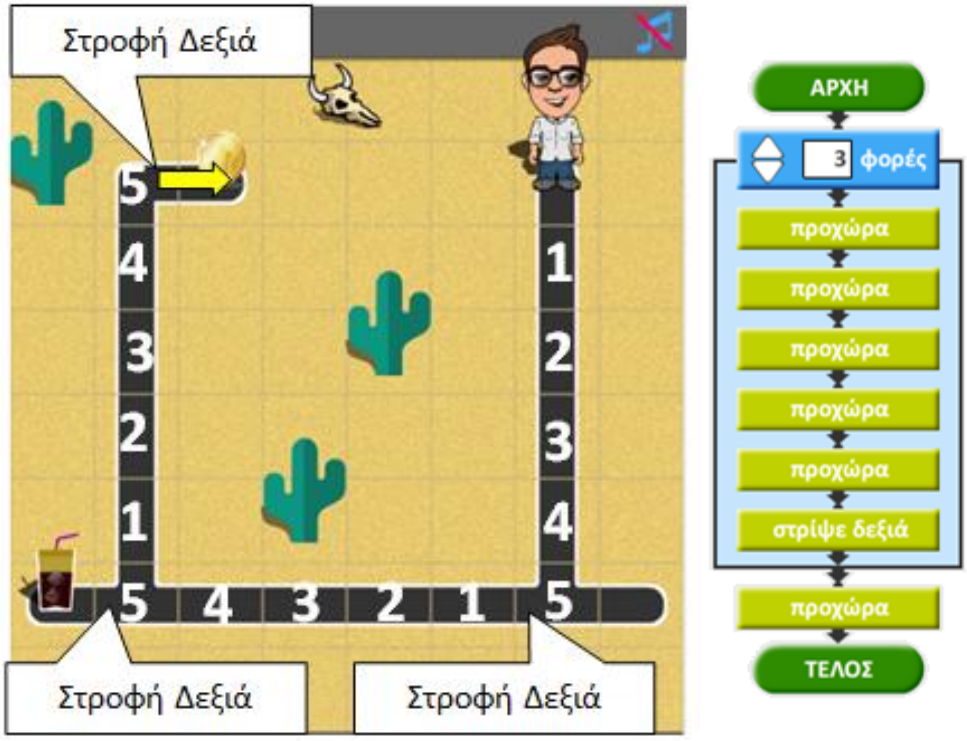
Σχήμα 3.36: Εικόνα παιχνιδιού και κώδικας σε γλώσσα C.

Όπως παρατηρούμε οι επαναλήψεις έχουν ρυθμιστεί έτσι ώστε να επαναλαμβάνουν 3 φορές το κάθε σύνολο κινήσεων σύμφωνα με τις απαιτήσεις της 5^{ης} πίστας. Όταν ολοκληρωθεί η πρώτη επανάληψη ο Θαλής θα βρίσκεται στη μέση της διαδρομής και αμέσως θα ξεκινήσει η δεύτερη επανάληψη που θα επαναλάβει 3 φορές το κάθε σύνολο κινήσεων του δεύτερου μισού της διαδρομής. Η τελευταία εντολή proxora(); εκτελείται για να οδηγήσει τον Θαλή στο νόμισμα.



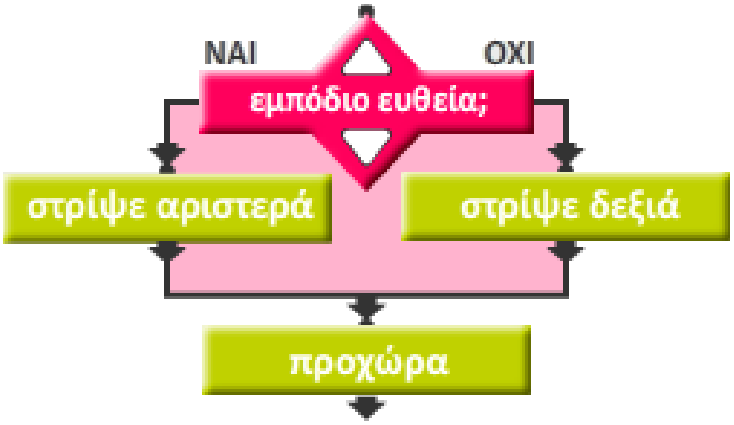
Στην 7η πίστα του παιχνιδιού μας γνωρίσαμε τη δομή της επιλογής και προγραμματίσαμε το Θαλή να επιλέγει μόνος του τη σωστή διαδρομή για το νόμισμα σύμφωνα με τα εμπόδια ή τα μονοπάτια που έβλεπε καθώς προχωρούσε προς το νόμισμα (Σχ. 3.37).





Σχήμα 3.37: Εικόνα παιχνιδιού και διάγραμμα ροής του κώδικα.

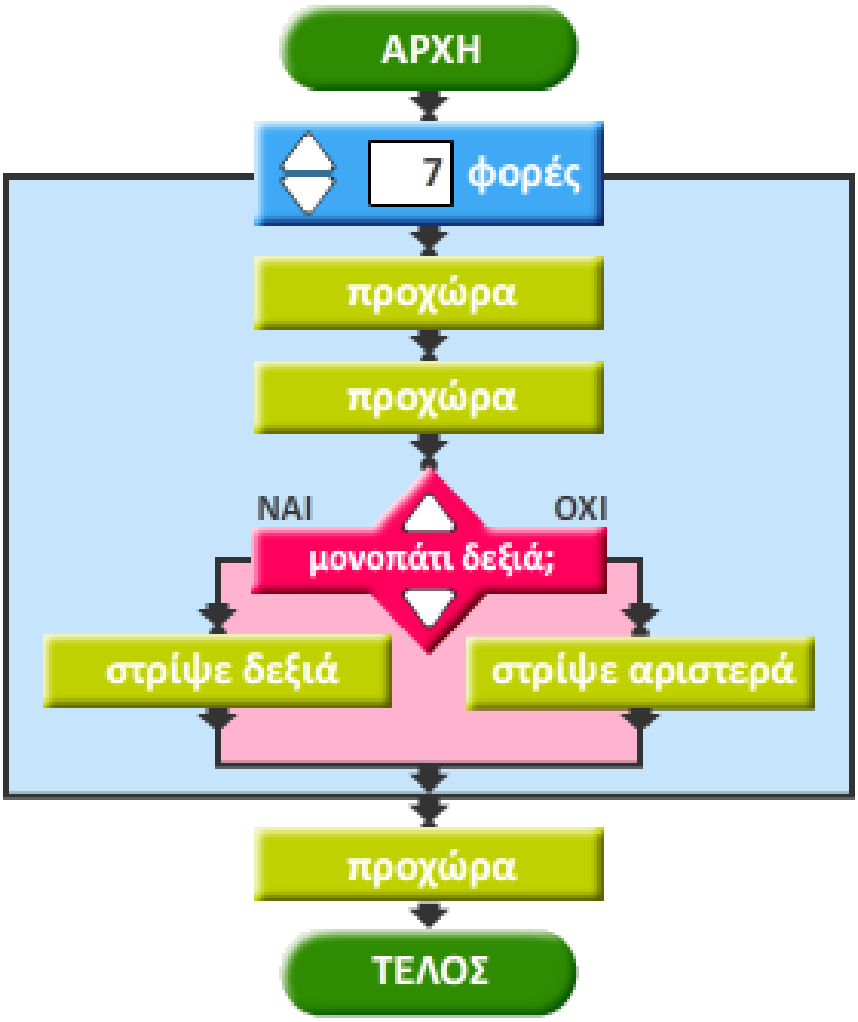
Γνωρίσαμε τη δομή της επιλογής μέσα από το διάγραμμα όπου μας δινόταν η ευκαιρία να επιλέξουμε ένα σύνολο επιλογών όπως, εμπόδιο ευθεία, εμπόδιο δεξιά ή αριστερά κ.τ.λ. Πώς όμως θα μπορούσαμε να γράψουμε σε κώδικα γλώσσας προγραμματισμού τη δομή της επιλογής; Στη συνέχεια παρουσιάζουμε την σύνταξη της δομής επιλογής σε μία γλώσσα προγραμματισμού (Σχ. 3.38).




```
if ( empodio_euthia() )
{
  strofi_aristera();
}
else
{
 strofi_dexia();
}
proxora();
```

Σχήμα 3.38: Διάγραμμα ροής και κώδικας σε γλώσσα C.

Η σύνταξη της δομής επιλογής ξεκινά με την λέξη **if** η οποία σημαίνει '**ΑΝ**', έπειτα ακολουθεί μία παρένθεση μέσα στην οποία τοποθετούμε τη συνθήκη που θέλουμε να ελέγξουμε η οποία μπορεί να ισχύει (ΝΑΙ) ή να μην ισχύει (ΟΧΙ). Στο παράδειγμα που μελετάμε ελέγχουμε αν υπάρχει εμπόδιο ευθεία για τον Θαλή (Σχ. 3.39). Στην περίπτωση που υπάρχει εμπόδιο ευθεία (ΝΑΙ) εκτελούμε τη διαδικασία «στρίψε αριστερά - strofi_aristera()» *αλλιώς* (**else**) εκτελούμε την διαδικασία «στρίψε δεξιά – strofi_dexia()».



Σχήμα 3.39: Διάγραμμα ροής του κώδικα.

Ας δούμε όμως το παράδειγμα της 7ης πίστας του παιχνιδιού μας και ας γράψουμε κώδικα γι' αυτό. Σε αυτό το παράδειγμα χρησιμοποιήσαμε μία δομή επανάληψης σε συνδυασμό με μία δομή επιλογής για να οδηγήσουμε τον Θαλή. Η δομή της επανάληψης είναι πλέον γνώριμη σε εμάς και αμέσως μπορούμε να καταλάβουμε από το διάγραμμα πως πρέπει να επαναλάβουμε 7 φορές την διαδικασία οδήγησης του Θαλή. Ας δούμε όμως τον κώδικα που συμπεριλαμβάνει και τη δομή επιλογής. Παρόμοια με τα προηγούμενα παραδείγματα η επανάληψη θα πρέπει να πραγματοποιηθεί 7 φορές. Μέσα σε αυτή την επανάληψη πραγματοποιούνται δύο διαδικασίες proxora(); και έπειτα γίνεται έλεγχος αν υπάρχει μονοπάτι στα δεξιά του Θαλή. Αν υπάρχει μονοπάτι τότε εκτελείται η διαδικασία strofi_dexia(); αλλιώς εκτελείται η διαδικασία strofi_aristera();

```
for (var fores_1 = 0;   fores_1 <7;   fores_1++)
{
  proxora();
  proxora();
  if ( monopati_dexia() )
  {
     strofi_dexia();
  }
  else
  {
    strofi_aristera();
  }
}
proxora();
```

Μπορείτε να χρησιμοποιήσετε τη δομή επιλογής για να οδηγήσετε τον Θαλή ένα βήμα πριν από το στο νόμισμα και να επιστρέψει στην αρχική του θέση; (Σχ. 3.40). Προσπαθήστε!

Σχήμα 3.40: Εικόνα παιχνιδιού.

3.10 Μία διαφορετική δομή επανάληψης με κώδικα

Στην 8η πίστα του παιχνιδιού μας γνωρίσαμε μία διαφορετική δομή επανάληψης που μας επέτρεπε να επαναλάβουμε μία διαδικασία μέχρι να πραγματοποιηθεί ένα γεγονός όπως το γεγονός που ο Θαλής παίρνει το νόμισμα.

Αυτή η δομή επανάληψης χρησιμοποιείται όταν δεν γνωρίζουμε ακριβώς πόσες φορές θέλουμε να επαναληφθεί μια διαδικασία. Ποιος είναι όμως ο κώδικας που κρύβεται πίσω από αυτή τη δομή επιλογής (Σχ. 3.41);

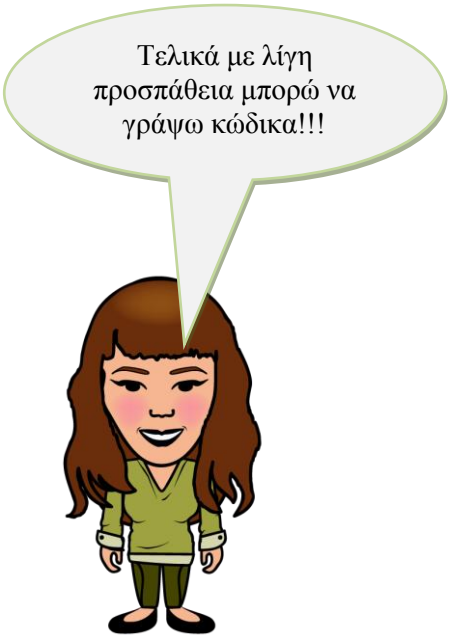
```
do
{

} while ( !pires_nomisma() );
```

Σχήμα 3.41: Διάγραμμα ροής και κώδικας σε γλώσσα C.

Η δομή επανάληψης που μελετάμε ξεκινά με την δεσμευμένη λέξη **do** που μεταφράζεται ως 'ΚΑΝΕ' και επαναλαμβάνει τις εντολές και τις διαδικασίες που βρίσκονται κλεισμένες μέσα στις αγκύλες όσο (**while**) ισχύει (ΝΑΙ) η συνθήκη που βρίσκεται στις παρενθέσεις.

Στο πιο πάνω παράδειγμα, που είναι απόσπασμα κώδικα από το παιχνίδι με τον Θαλή, όποια διαδικασία τοποθετήσουμε ανάμεσα στις αγκύλες θα επαναλαμβάνεται όσο (while) ΔΕΝ ΙΣΧΥΕΙ (!) η συνθήκη pires_nomisma(), δηλαδή όσο ο Θαλής δεν έχει πάρει το νόμισμα.

Η διαδικασία pires_nomisma() όταν εκτελείται μας δίνει μία τιμή ΝΑΙ ή ΟΧΙ αν ο Θαλής έχει πάρει το νόμισμα ή όχι



αντίστοιχα. Το σύμβολο του θαυμαστικού ! που χρησιμοποιούμε μπροστά από τη διαδικασία pires_nomisma() σημαίνει ΟΧΙ και όταν τοποθετείται μπροστά από μία διαδικασία δημιουργεί μία συνθήκη άρνησης. Έτσι η ακόλουθη γραμμή κώδικα σημαίνει:

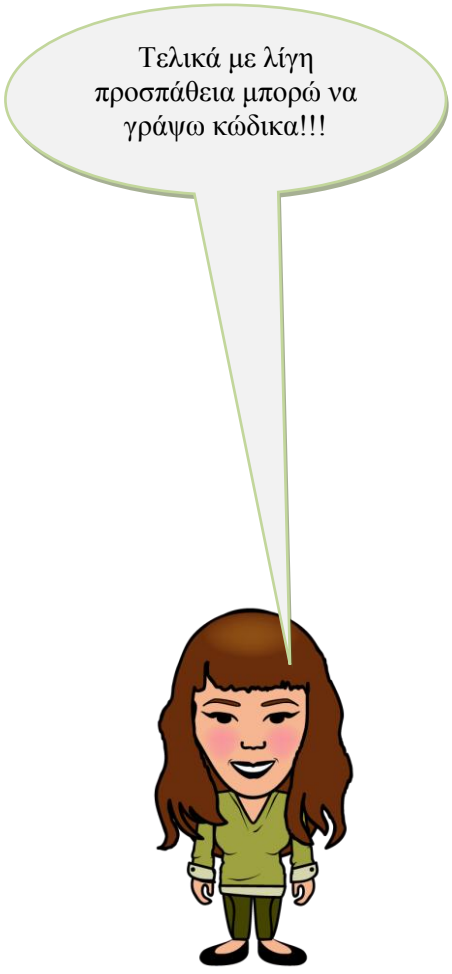
!pires_nomisma() → ΔΕΝ ΠΗΡΕΣ ΤΟ ΝΟΜΙΣΜΑ

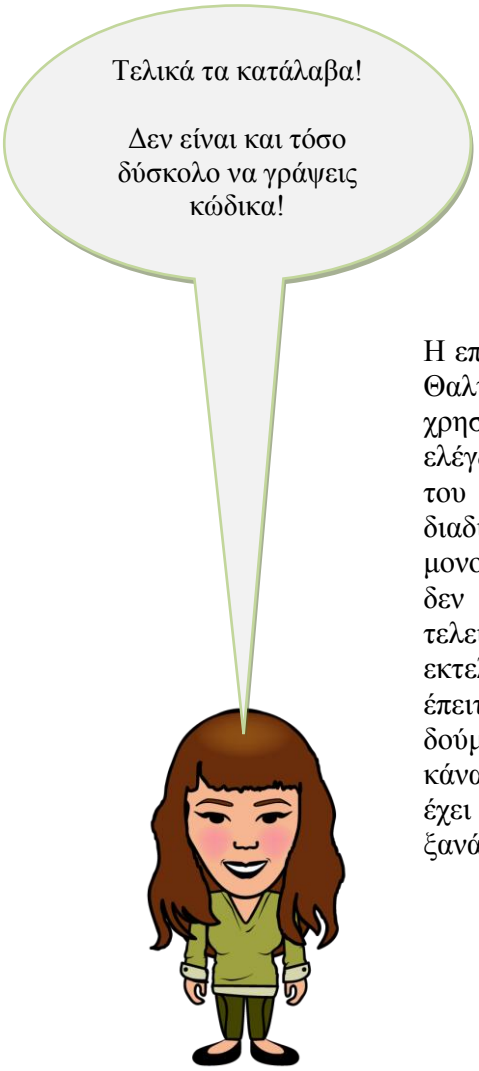
Και η ακόλουθη γραμμή κώδικα σημαίνει:

while (!pires_nomisma()); → ΟΣΟ (ΔΕΝ ΠΗΡΕΣ ΤΟ ΝΟΜΙΣΜΑ)

Αφού καταλάβαμε τη λειτουργία της νέας δομής επανάληψης do, μπορούμε να τη χρησιμοποιήσουμε για να καταλάβουμε τον κώδικα της 8ης πίστας του παιχνιδιού μας (Σχ. 3.42, Σχ.3.43).

Σχήμα 3.42: Διάγραμμα ροής του κώδικα.






```
do
{
  if ( monopati_aristera() )
 {
   strofi_aristera();
 }
 else
 {

 }
 proxora();
} while ( !pires_nomisma()
);
```

Σχήμα 3.43: Κώδικας σε γλώσσα C.

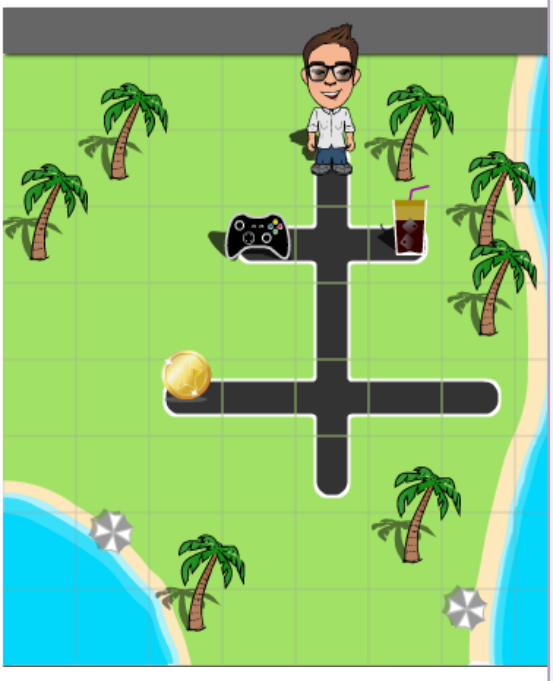
Η επανάληψη εκτελείται συνεχώς έως ότου ο Θαλής πάρει το νόμισμα. Σε κάθε επανάληψη χρησιμοποιούμε τη δομή επιλογής if για να ελέγξουμε αν υπάρχει μονοπάτι στα αριστερά του Θαλή και αν υπάρχει εκτελούμε τη διαδικασία strofi_aristera();. Αν δεν υπάρχει μονοπάτι στα αριστερά του Θαλή (else) τότε δεν εκτελούμε καμία ενέργεια. Αφού τελειώσει ο έλεγχος της δομής επιλογής εκτελούμε μία διαδικασία proxora(); και έπειτα ελέγχουμε τη συνθήκη while για να δούμε αν ο Θαλής μετά από τις ενέργειες που κάναμε πήρε το νόμισμα. Αν ο Θαλής δεν έχει πάρει το νόμισμα επαναλαμβάνουμε ξανά όλη τη διαδικασία που περιγράψαμε.

Άσκηση 1

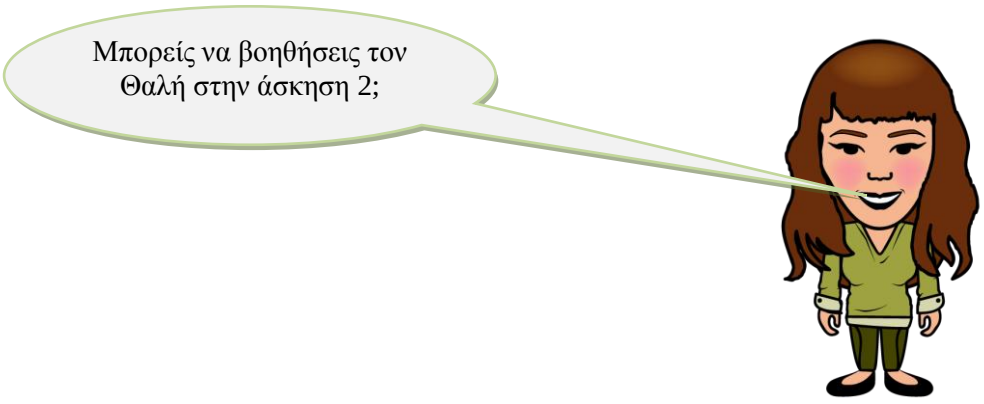
Να γράψετε κώδικα για την πίστα του σχήματος ώστε ο Θαλής:

1. Να πάει για ένα καφεδάκι.
2. Να παίξει ένα ηλεκτρονικό παιχνίδι.

Αφού γράψετε τον κώδικα με διάγραμμα μπορείτε να γράψετε και κώδικα με κείμενο;



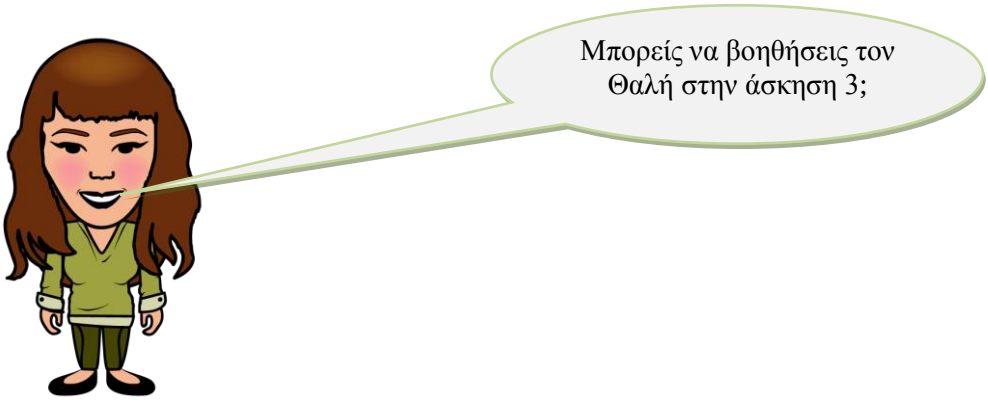
Γράψε εδώ τον κώδικα σου



Άσκηση 2

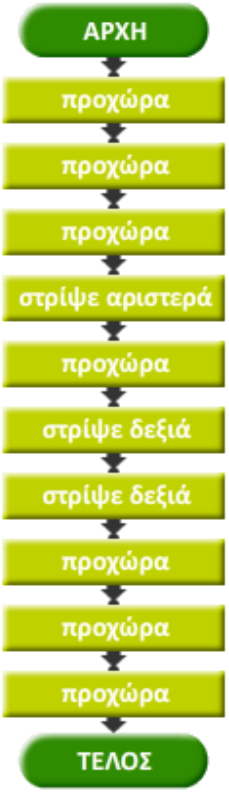
Μπορείτε να γράψετε κώδικα για την πίστα του σχήματος ώστε ο Θαλής να φτάσει ένα βήμα πριν το νόμισμα και έπειτα να επιστρέψει στη θέση που βρίσκεται ξανά; Θεωρήστε ότι ο Θαλής ξεκινά από το σημείο που βλέπετε στην παρακάτω εικόνα και γράψετε κώδικα με κείμενο.

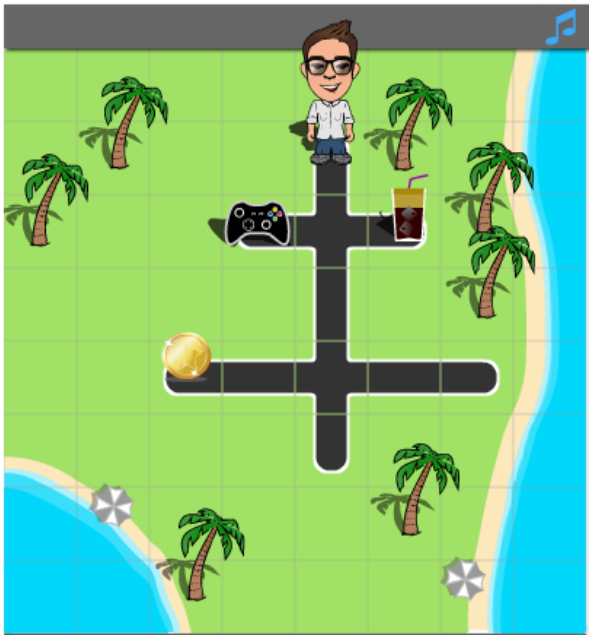
Γράψε εδώ τον κώδικα σου

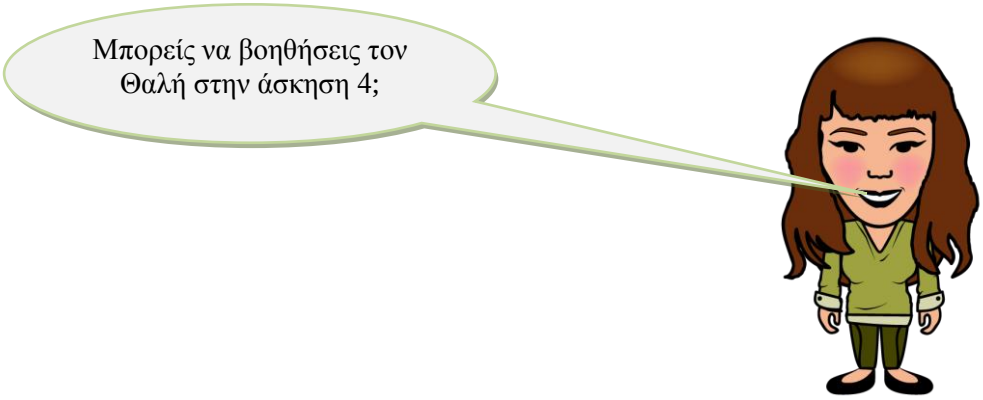


Άσκηση 3

Μπορείτε να σημειώσετε στην πίστα του σχήματος ποια θα είναι η διαδρομή που θα κάνει ακριβώς ο Θαλής;



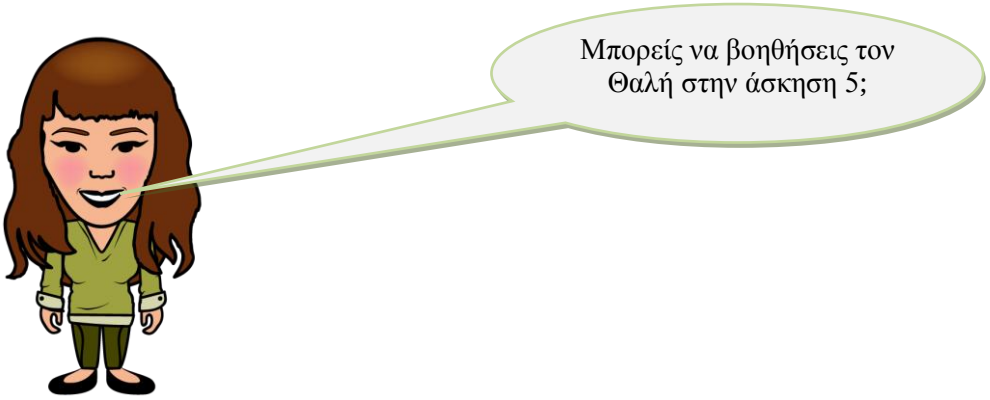




Άσκηση 4

Ο κώδικας που βλέπετε στο σχήμα θα μπορούσε να βελτιωθεί ώστε να μην χρησιμοποιεί τόσες εντολές μέσα στην επανάληψη. Μπορείτε να γράψετε ένα πιο μικρό κώδικα για αυτό το παράδειγμα που να οδηγεί τον Θαλή στο νόμισμα;

Γράψε εδώ τον κώδικα σου



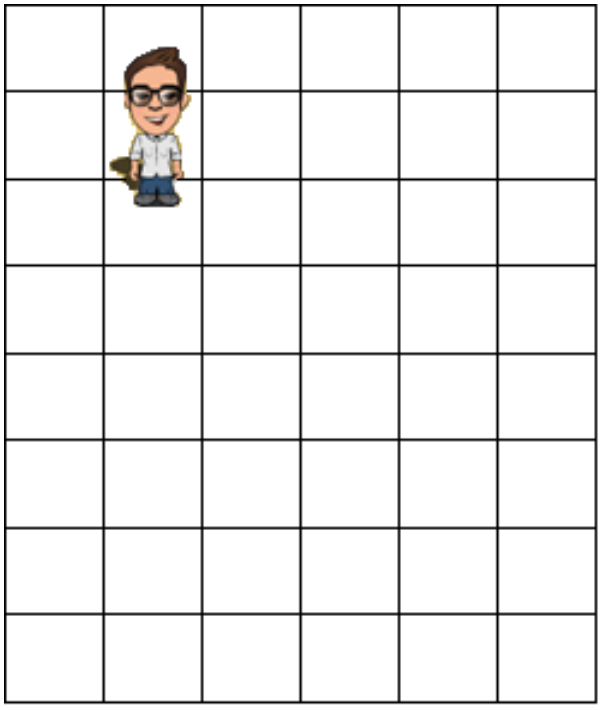
Άσκηση 5

Γράψτε κώδικα με κείμενο που να οδηγεί τον Θαλή στο ποτήρι με τον καφέ.

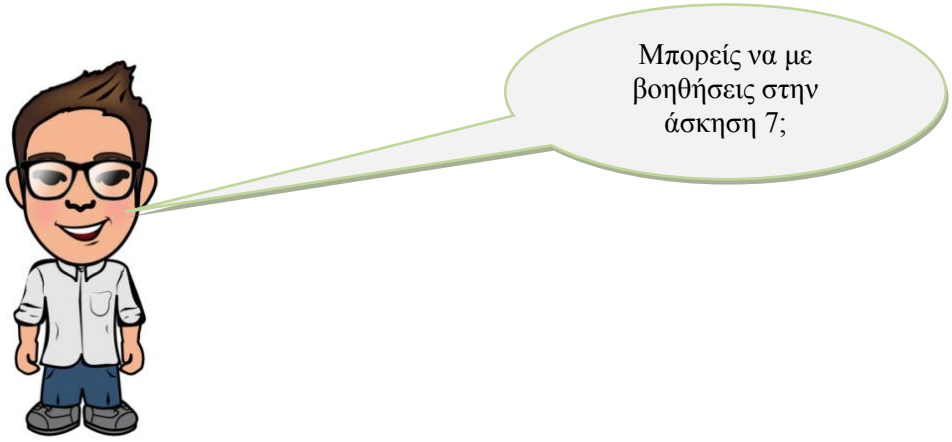
Γράψε εδώ τον κώδικα σου

Άσκηση 6

Μπορείτε να σχεδιάσετε μία διαδρομή που επιθυμείτε στο ακόλουθο σχήμα και έπειτα να γράψετε κώδικα που θα οδηγεί τον Θαλή στο νόμισμα; Τοποθετήστε το νόμισμα στο σημείο που εσείς επιθυμείτε.

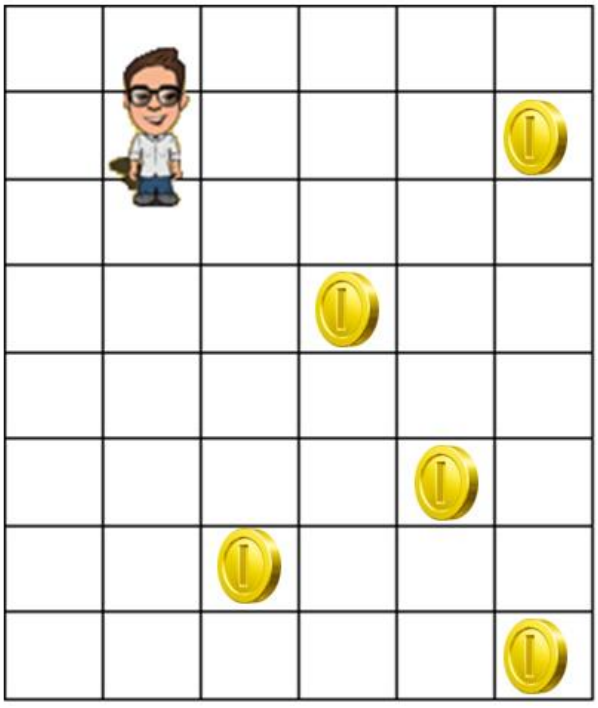


Γράψε εδώ τον κώδικα σου



Άσκηση 7

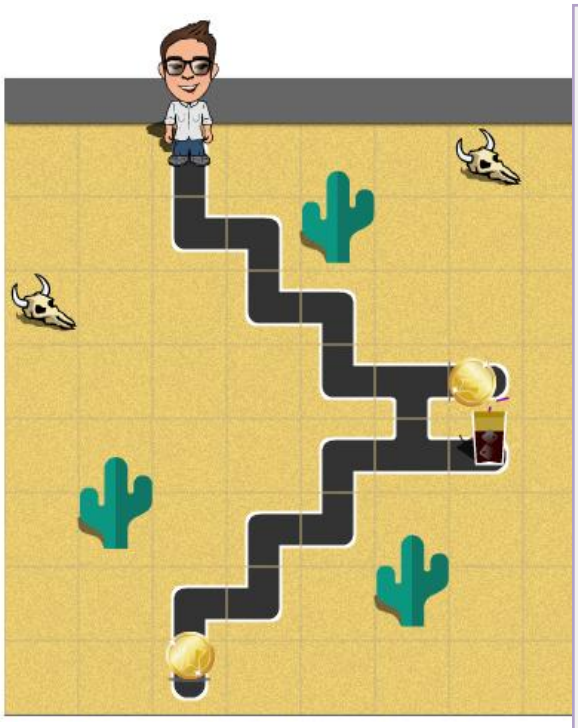
Σχεδιάστε και προγραμματίστε μία διαδρομή για τον Θαλή έτσι ώστε να συλλέξει όλα τα νομίσματα που βρίσκονται μέσα στην ακόλουθη πίστα.



Γράψε εδώ τον κώδικα σου

Άσκηση 8

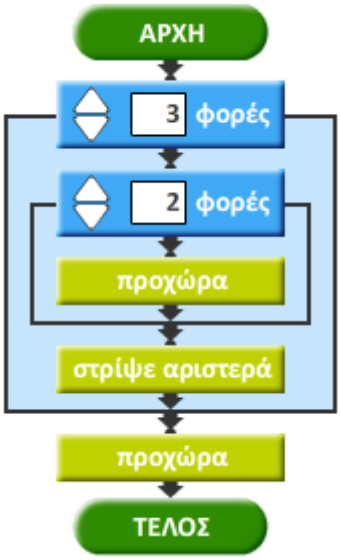
Μπορείτε να γράψετε κώδικα για την ακόλουθη πίστα ώστε ο Θαλής να πάρει και τα δύο νομίσματα;

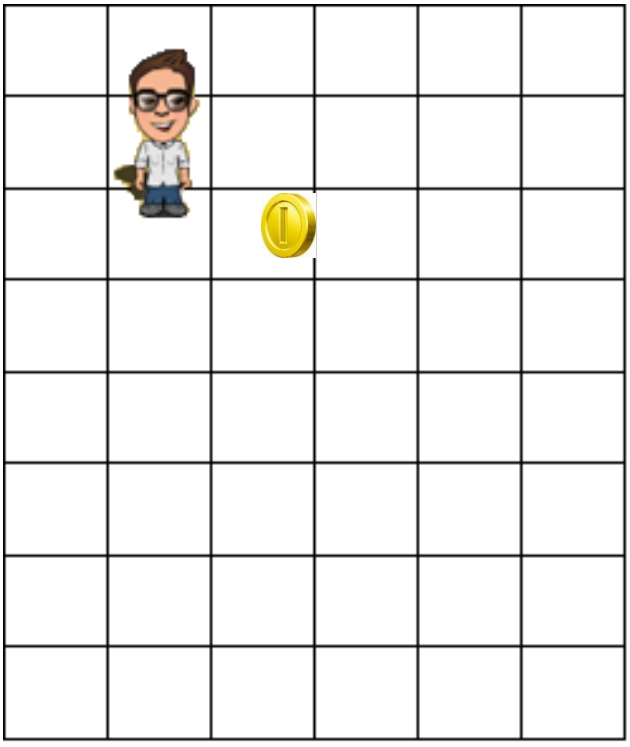


Γράψε εδώ τον κώδικα σου

Άσκηση 9

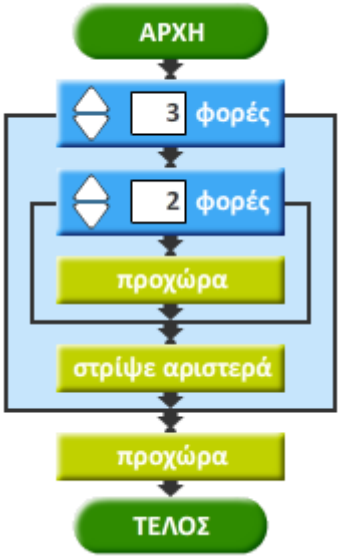
Σχεδιάστε τη διαδρομή που θα εκτελέσει ο ακόλουθος κώδικας.



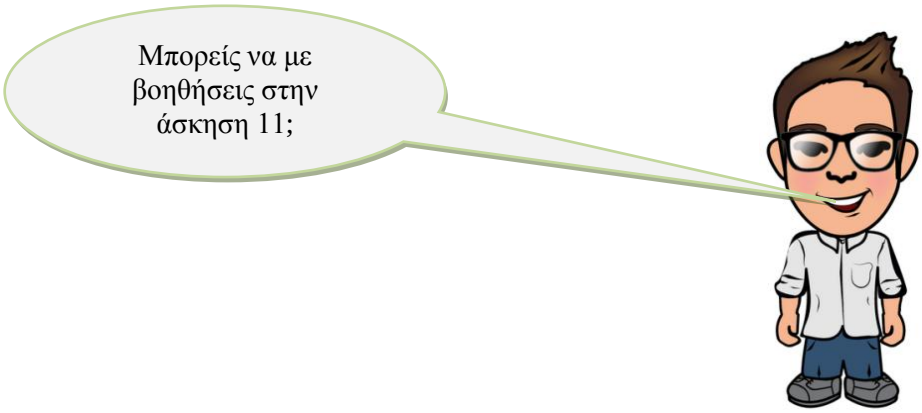


Άσκηση 10

Γράψτε κώδικα με κείμενο για το ακόλουθο διάγραμμα.

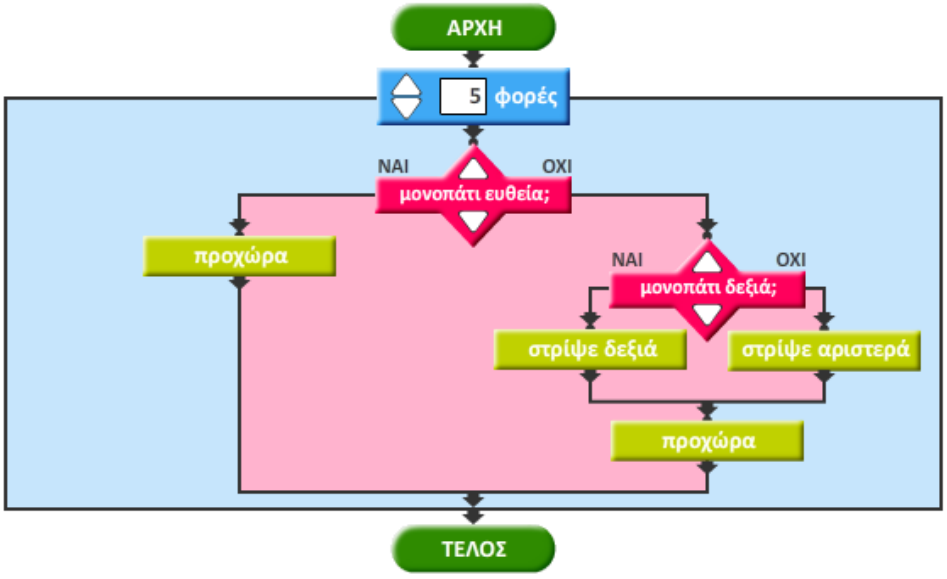


Γράψε εδώ τον κώδικα σου

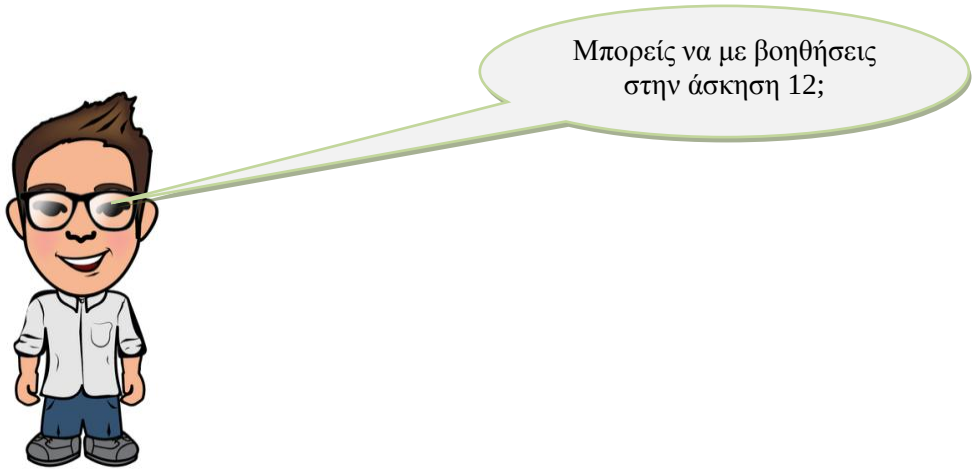


Άσκηση 11

Γράψτε κώδικα με κείμενο για το ακόλουθο διάγραμμα.



Γράψε εδώ τον κώδικα σου

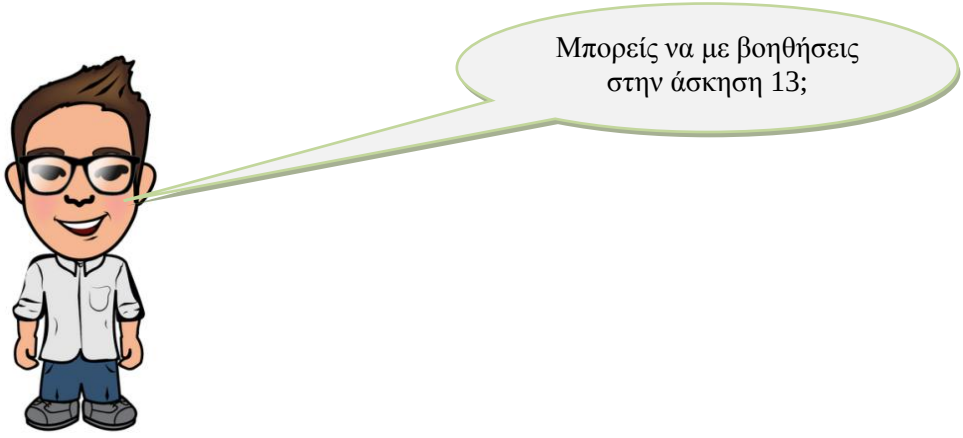


Άσκηση 12

Μπορείτε να συμπληρώσετε τον πίνακα τιμών για τον ακόλουθο κώδικα;

```
for (var fores_1 = 0;   fores_1 < 3;   fores_1++)
{
  proxora();
  strofi_aristera();
  proxora();
  strofi_dexia();
}
for (var fores_2 = 0;   fores_2 < 3;   fores_2++)
{
  proxora();
  strofi_dexia();
  proxora();
  strofi_aristera();
}
proxora();
```

Πίνακας Τιμών							
Επανάληψη							
Τιμή fores_1							
fores_1 < 3 ;							
Τιμή fores_2							
fores_2 < 3 ;							

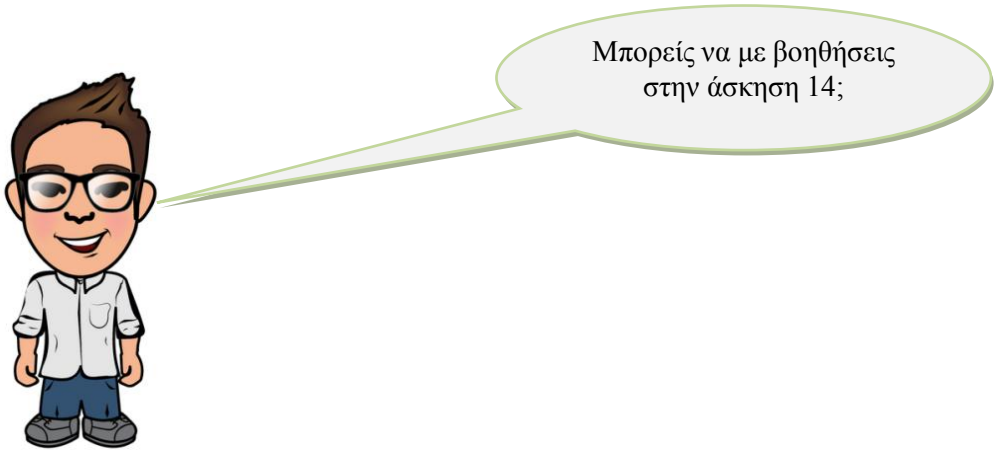


Άσκηση 13

Σχεδιάστε το διάγραμμα ροής για το ακόλουθο τμήμα κώδικα.

```
do
{
  for (var fores_1 = 1;   fores_1 <10;   fores_1++)
  {
    proxora();
    strofi_dexia();
    for (var fores_2 = 0;   fores_2 < 3;   fores_2++)
    {
      proxora();
      strofi_aristera();
   }
  }
}while(!pires_nomisma);
```

Σχεδίασε εδώ το διάγραμμα ροής του κώδικα



Άσκηση 14

Μπορείτε να βρείτε τα συντακτικά λάθη στον ακόλουθο κώδικα;

```
do
{
  for (var fores-1 = 0   fores_1 <10;   fores_1++)
  {
    proxora);
    strofi-dexia();
    stroφι_aristera();
    proxora()
  }
}while(!pires_nomisma)
```

Σημείωσε τα λάθη

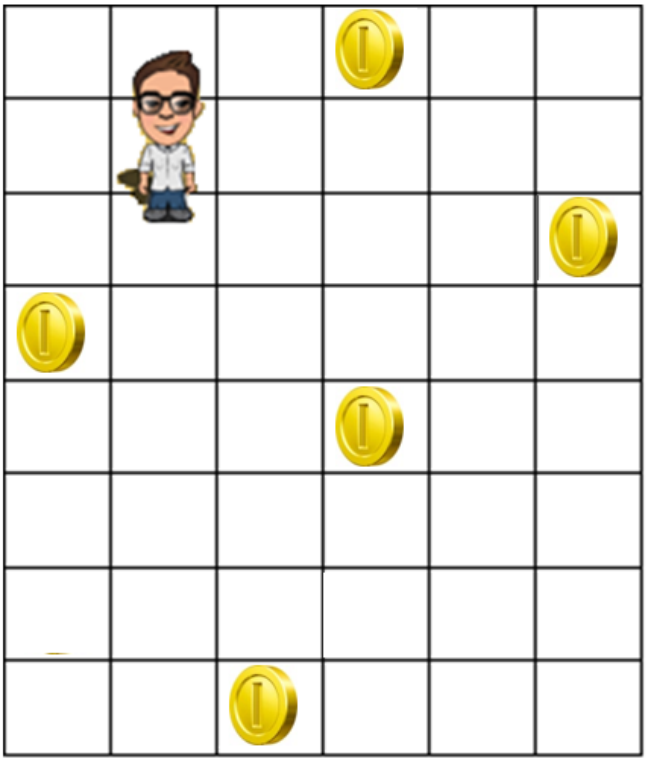
Άσκηση 15

Μπορείτε να γράψετε κώδικα για την ακόλουθη πίστα ώστε ο Θαλής να φτάσει ένα βήμα πριν το νόμισμα και να επιστρέψει ξανά πίσω στην αρχική του θέση;

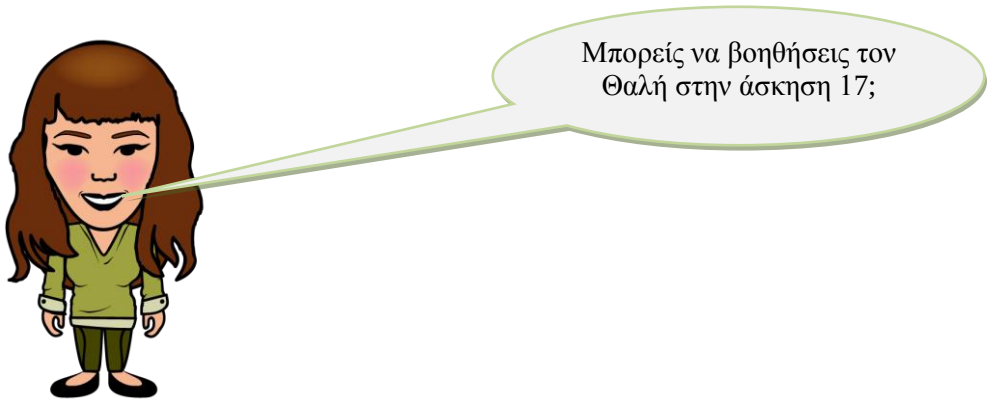
Γράψε εδώ τον κώδικα σου

Άσκηση 16

Μπορείτε να σχεδιάσετε το διάγραμμα ροής ώστε προγραμματίζωντας τον Θαλή να μαζέψει όλα τα νομίσματα εκτελωντας τη συντομότερη δυνατή διαδρομή;

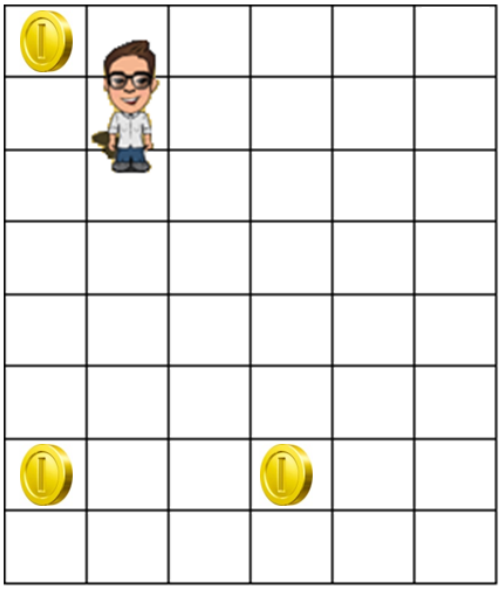


Σχεδίασε εδώ το διάγραμμα ροής του κώδικα



Άσκηση 17

Μπορείτε να γράψετε κώδικα ώστε προγραμματίζωντας τον Θαλή να μαζέψει όλα τα νομίσματα εκτελωντας τη συντομότερη δυνατή διαδρομή;



Γράψε εδώ τον κώδικα σου

4

Γνωρίζοντας προγραμματιστικά εργαλεία

4.1 Προγραμματίζοντας με το MSKodu

Το MSKodu αναπτύχθηκε από την Microsoft (Microsoft Kodu Game Lab) ως μια οπτική γλώσσα προγραμματισμού με σκοπό να εισάγει τον ενδιαφερόμενο στον κόσμο του προγραμματισμού μέσω της δημιουργίας παιχνιδιών (Σχήμα 4.1). Μέσα από το προγραμματιστικό περιβάλλον του MSKodu επιτρέπεται στο χρήστη να δημιουργεί χαρακτήρες και γραφικά και να περιηγείται σε προγραμματιζόμενους τρισδιάστατους κόσμους. Στο MSKodu δυο στοιχεία κυριαρχούν κατά τη δημιουργία ενός παιχνιδιού:

- ο κόσμος, δηλαδή η πίστα του παιχνιδιού, κάθε μορφή εικόνας μέσα στην οποία ο ήρωας κινείται, και
- τα αντικείμενά του, όπου ως αντικείμενο ορίζουμε κάθε στοιχείο μέσα στο παιχνίδι το οποίο έχει καθορισμένη συμπεριφορά.

Για να γίνουν κατανοητά οι παραπάνω ορισμοί αναφέρουμε το παράδειγμα ενός διαστημικού παιχνιδιού στο οποίο το διάστημα είναι ο κόσμος του προγράμματος. Αν δημιουργήσουμε έναν ήρωα στο MSKodu, μέσα σε ένα διαστημόπλοιο, ο οποίος όταν δει έναν αστεροειδή να εκτοξεύει έναν πύραυλο προς αυτόν, τότε θα έχουμε δώσει μια συμπεριφορά στο αντικείμενο διαστημόπλοιο.

Σχήμα 4.1: Παράδειγμα οθόνης παιχνιδιού με το MSKodu.

Για να μπορεί ο χρήστης να αλληλεπιδρά με το αντικείμενο ο προγραμματιστής διαχειρίζεται τους παρακάτω αισθητήρες:

- ✓ Ο αισθητήρας πληκτρολόγιο (keyboard) μέσω του οποίου τα αντικείμενα μπορούν να αντιδρούν στη χρήση του πληκτρολογίου από το χρήστη.
- ✓ Ο αισθητήρας ποντίκι (mouse) μέσω του οποίου τα αντικείμενα μπορούν να αντιδρούν στις κινήσεις του ποντικιού.
- ✓ Ο αισθητήρας χειριστήριο (gamepad) μέσω του οποίου τα αντικείμενα μπορούν να αντιδρούν μέσω της κονσόλας του X-BOX

Κάθε αντικείμενο περιλαμβάνει ένα σύνολο ιδιοτήτων και ενεργειών οι οποίες προγραμματίζονται και καθορίζουν την συμπεριφορά του ήρωα. Μερικές από τις πιο συνηθισμένες ιδιότητες και ενέργειες που συναντάμε κατά τον προγραμματισμό είναι:

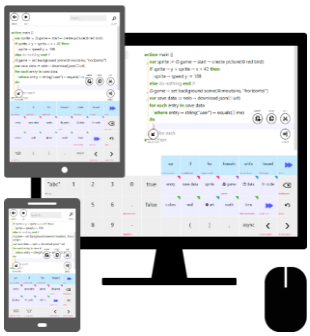
- ✓ Βλέπω (See)
- ✓ Ακούω (Hear)
- ✓ Πέφτω Πάνω (Bump)
- ✓ Πετυχαίνω (Shot Hit)
- ✓ Έχω (Got)

- ✓ Κουβαλιέμαι (Held By)
- ✓ Τρώω (Eat)
- ✓ Εξαφανίζω (Vanish)
- ✓ Εκτινάζω (Launch)
- ✓ Αρπάζω (Grab)

Μπορείτε να ξεκινήσετε την εμπειρία σας με το MSKodu από την παρακάτω επίσημη διεύθυνση:

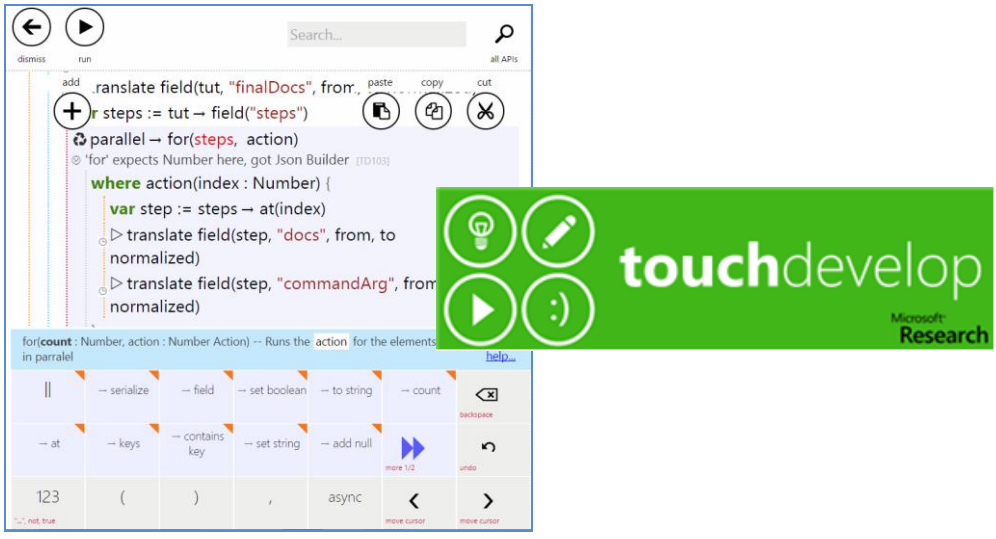
http://www.kodugamelab.com/

4.2 Προγραμματίζοντας με το TouchDevelop



Το TouchDevelop χρησιμοποιείται για την ανάπτυξη εφαρμογών κυρίως σε κινητά τηλέφωνα και ταμπλέτες μέσω προγραμματισμού.

Ο προγραμματισμός σε περιβάλλον TouchDevelop πραγματοποιείται με χρήση εντολών που περιλαμβάνονται σε βιβλιοθήκες, όπου ο προγραμματιστής με επαφή (touch) εισάγει αυτές στον κώδικα. Στο Σχήμα 4.2 απεικονίζεται η οθόνη ενός κινητού τηλεφώνου στο οποίο ο χρήστης επιλέγει εντολή κατά τον προγραμματισμό της εφαρμογής του.



Σχήμα 4.2: Οθόνη κατά την ανάπτυξη εφαρμογής με το TouchDevelop.

Μπορείτε να ξεκινήσετε την εμπειρία σας με το TouchDevelop από την παρακάτω επίσημη διεύθυνση:

https://www.touchdevelop.com/

4.3 Η πηγή του DreamSpark

Το DreamSpark αποτελεί μια προσφορά της Microsoft μέσω τις οποίας ο κάθε προγραμματιστής μπορεί να αποκτήσει δωρεάν λογισμικά για την ανάπτυξη των εφαρμογών του. Μερικά από τα διαθέσιμα λογισμικά που μπορεί ένας φοιτητής ή ένα μέλος ακαδημαϊκού ιδρύματος, να προμηθευτεί μέσω του DreamSpark για ερευνητικούς και εκπαιδευτικούς σκοπούς είναι τα παρακάτω:

- Visual Studio 2008
- Windows Server 2008 Standard
- SQL Server 2008 Developer
- Microsoft Expression Studio 2
- XNA Game Studio 2
- IT Academy Student Pass
- Visual Studio 2005 Professional
- Virtual PC
- Microsoft Visual Studio 2008 Express Editions
- Microsoft Visual Studio 2005 Express Editions
- Visual C#

Μπορείτε να ξεκινήσετε την εμπειρία σας με το DreamSpark από την παρακάτω επίσημη διεύθυνση:

https://www.dreamspark.com/

4.4 Προγραμματίζοντας με το Windows App Studio

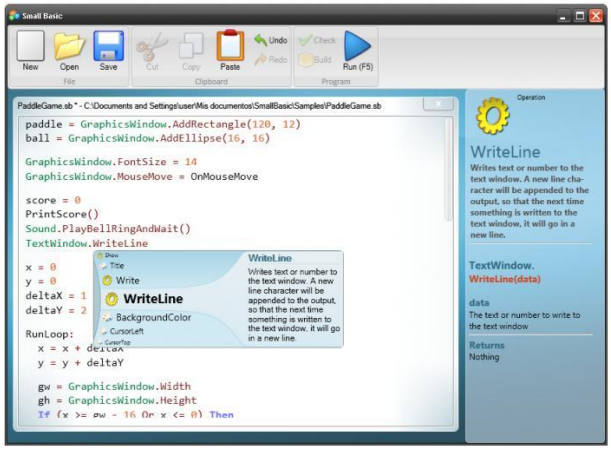
Το Windows App Studio αποτελεί εργαλείο δημιουργίας εφαρμογών για Windows Phone εφαρμογές. Δυναμική του Windows App Studio αποτελεί το γεγονός ότι μπορεί να παράγει κώδικα για το εργαλείο Visual Studio.

Μπορείτε να ξεκινήσετε την εμπειρία σας με το Windows App Studio από την παρακάτω επίσημη διεύθυνση:

http://appstudio.windows.com/en-us

4.5 Προγραμματίζοντας με το Small Basic

Το Microsoft Small Basic προσφέρει ένα φιλικό περιβάλλον με το οποίο είναι πολύ εύκολο να αναπτύξετε μια προγραμματιστική εφαρμογή. Για το σκοπό αυτό το Small Basic διαθέτει βιβλιοθήκες ώστε να αναπτύξετε σε πολύ γρήγορο χρόνο την εφαρμογή σας (Σχ. 4.3).



Σχήμα 4.3: Περιβάλλον Small Basic.

Μπορείτε να ξεκινήσετε την εμπειρία σας με το Small Basic από την παρακάτω επίσημη διεύθυνση:

http://smallbasic.com/

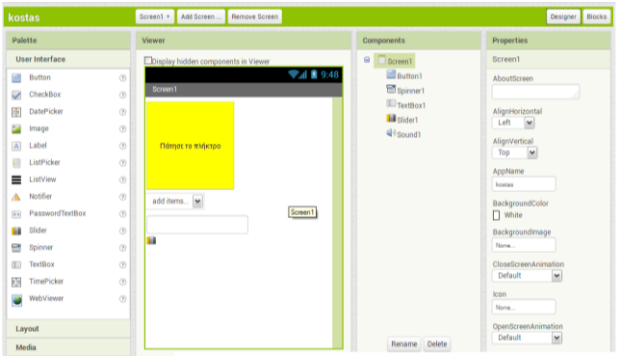
4.6 Προγραμματίζοντας με το App Inventor

Το App Inventor για Android αποτελεί περιβάλλον οπτικού προγραμματισμού και υποστηρίζεται από το MIT. Με χρήση βιβλιοθηκών του App Inventor μπορούμε να σχεδιάσουμε και να αναπτύξουμε την εφαρμογή μας σε γρήγορο χρόνο. Το App Inventor για την ευκολία του προγραμματισμού χρησιμοποιεί περιβάλλον αλληλεπίδρασης, παρόμοιο με αυτό του Scratch. Για την ανάπτυξη των εφαρμογών στο προγραμματιστικό εργαλείο συναντάμε:

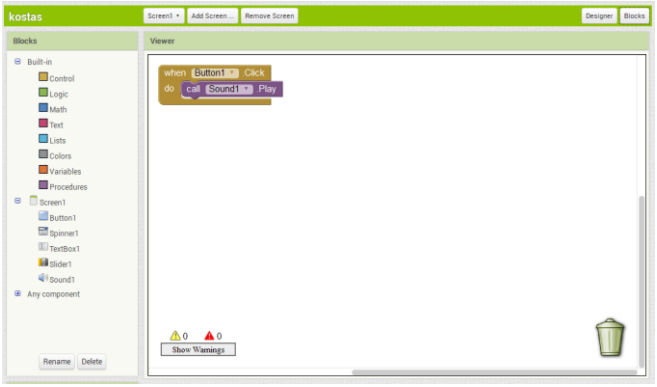
- το App Inventor Designer στο οποίο σχεδιάζουμε την εφαρμογή όπως αυτή θα απεικονίζεται στο κινητό του χρήστη, και

- το App Inventor Blocks Editor μέσω του οποίου, με μορφή πλακιδίων αναπτύσσουμε τον κώδικα.

Με την ολοκλήρωση της εφαρμογής, ακολουθούμε διαδικασία στην οποία το αρχείο τύπου .apk μπορεί να εισαχθεί σε συσκευές Android (Σχ. 4.4).



(α)



(β)

Σχήμα 4.4: Περιβάλλον ανάπτυξης α) γραφικών και β) κώδικα με μορφή πλακιδίων.

Μπορείτε να ξεκινήσετε την εμπειρία σας με το App Inventor από την παρακάτω επίσημη διεύθυνση:

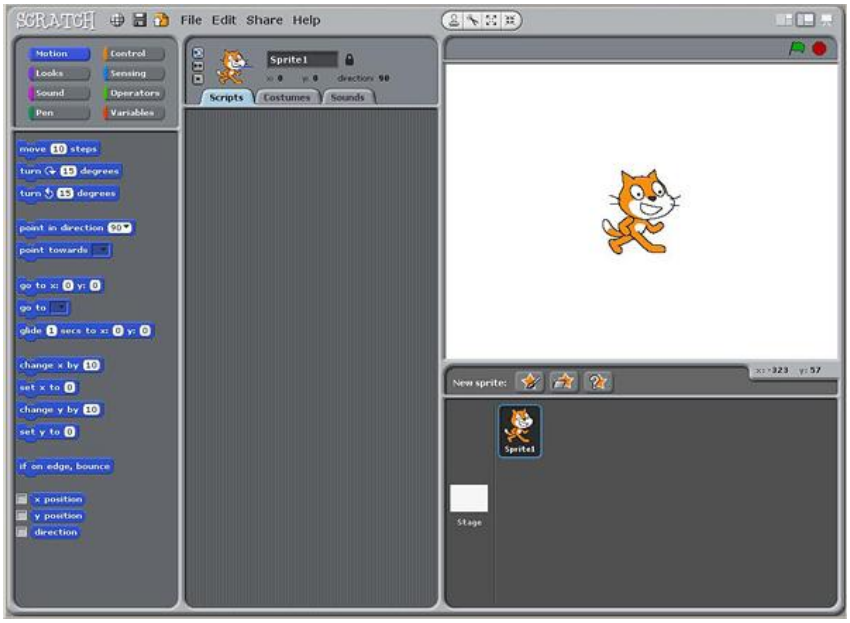
http://appinventor.mit.edu/explore/

4.7 Προγραμματίζοντας με το Scratch

Το Scratch είναι μια οπτική γλώσσα προγραμματισμού σε Squeak. Το Scratch έχει αναπτυχθεί από μια μικρή ομάδα ερευνητών στο Lifelong Kindergarten Group στο MIT Media Lab (Σχήμα 4.5). Ο προγραμματισμός του Scratch γίνετε με χρήση πλακιδίων (Σχήμα 4.6). Για τη δημιουργία των έργων (project) στο Scratch χρησιμοποιούμε αντικείμενα τα οποία καλούντα μορφές (sprite). Οι μορφές έχουν ιδιότητες τις οποίες μπορούμε να τις τροποποιήσουμε. Για την κίνηση των μορφών χρησιμοποιούμε εντολές (πλακίδια) με τα οποία δημιουργούμε το σενάριο του έργου μας.

Μπορείτε να ξεκινήσετε την εμπειρία σας με το Scratch από την παρακάτω επίσημη διεύθυνση:

https://scratch.mit.edu/



Σχήμα 4.5: Περιβάλλον του λογισμικού Scratch.

Σχήμα 4.6: Παράδειγμα εντολών κώδικα με μορφή πλακιδίων.

4.8 Βουτώντας στο βυθό των υπολογιστικών συστημάτων

Έπειτα από τη ραγδαία ανάπτυξη της επιστήμης των υπολογιστών, αυξήθηκε σε μεγάλο βαθμό και ο αριθμός των ατόμων που ασχολούνται με τη μελέτη της λειτουργίας τους. Τα υπολογιστικά συστήματα σήμερα αποτελούν αντικείμενο μελέτης ακόμα και για μικρά παιδιά. Ολοένα και περισσότερο παρατηρείται το φαινόμενο της ενασχόλησης των παιδιών με την επιστήμη των υπολογιστών ακόμα και στα βαθιά νερά της πληροφορικής.

Η περιέργεια για αναζήτηση στο βάθος των υπολογιστικών συστημάτων και η ανάπτυξη της τεχνολογίας των ολοκληρωμένων κυκλωμάτων έφερε στην επιφάνεια ένα σύνολο από εκπαιδευτικές πλατφόρμες οι οποίες δίνουν τη δυνατότητα ακόμα και σε μικρά παιδιά να ασχοληθούν με την μελέτη και ανάπτυξη ενσωματωμένων συστημάτων και υπολογιστικών συστημάτων γενικότερα.

Αυτές οι εκπαιδευτικές πλατφόρμες χρησιμοποιούν μεθόδους προγραμματισμού και σύνδεσης υλικού που δεν απαιτούν πλέον εξειδικευμένες γνώσεις από έμπειρους μηχανικούς. Μπορούμε πλέον να χρησιμοποιούμε τις εκπαιδευτικές πλατφόρμες με ευκολία και να αναπτύσσουμε ενσωματωμένα πολογιστικά συστήματα τα οποία εκτελούν ένα σύνολο από λειτουργίες που συναντάμε ακόμα και στα πιο εξειδικευμένα συστήματα αυτομάτου ελέγχου. Στη συνέχεια αναφέρουμε περιληπτικά βασικές πλατφόρμες ανάπτυξης που χρησιμοποιούνται σήμερα στον τομέα της πληροφορικής και των ενσωματωμένων συστημάτων.

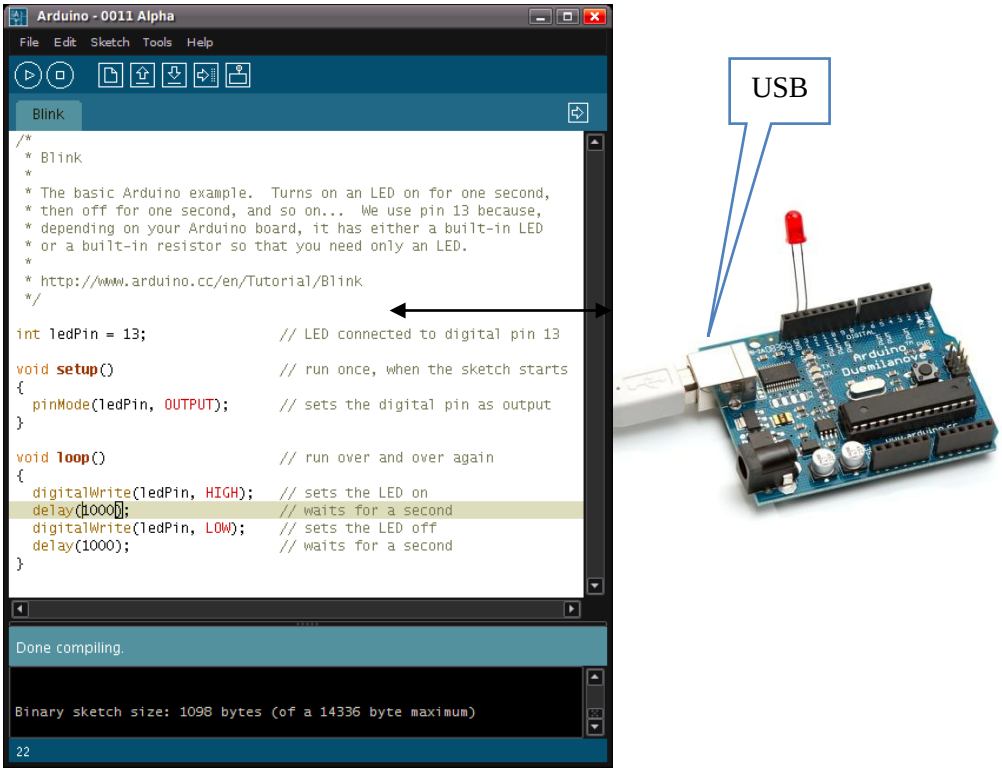
4.9 Η πλατφόρμα Arduino

Η πλατφόρμα Arduino είναι μία εκπαιδευτική πλατφόρμα με αντικείμενο τον προγραμματισμό ενσωματωμένων συστημάτων που χαρακτηρίζεται από ένα μεγάλο βαθμό ευκολίας και χρηστικότητας. Αυτή η πλατφόρμα υποστηρίζει ένα σύνολο έτοιμων εκπαιδευτικών μονάδων (πλακέτες) χαμηλού κόστους που καλύπτουν ένα μεγάλο σύνολο εφαρμογών στον κόσμο των ενσωματωμένων συστημάτων. Η πλατφόρμα Arduino χρησιμοποιεί τη γλώσσα προγραμματισμού Wiring C που βασίζεται στην γλώσσα προγραμματισμού C++.

Η γλώσσα αυτή είναι κατάλληλα σχεδιασμένη ώστε να παρέχει ευκολία στο χρήστη καθώς χρησιμοποιεί έτοιμες βιβλιοθήκες που μειώνουν στο ελάχιστο την διαδικασία οδήγησης περιφερειακών μονάδων σε ένα ενσωματωμένο σύστημα (Σχ. 4.7). Ο χρήστης χρειάζεται βασικές γνώσεις προγραμματισμού και λίγο χρόνο να εξοικειωθεί με τις βιβλιοθήκες και τη χρήση τους ώστε να προγραμματίσει ένα ενσωματωμένο σύστημα. Η πλατφόρμα μας παρέχει ένα σύνολο από έτοιμες ηλεκτρονικές πλακέτες ανάπτυξης που διαθέτουν επεξεργαστές έτοιμους για προγραμματισμό.

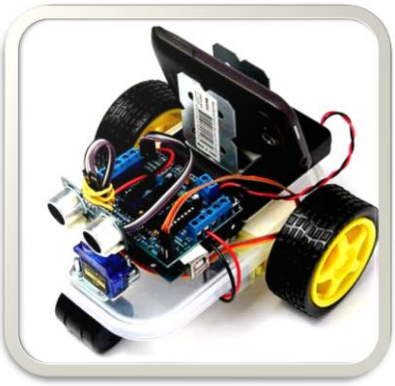
Η διαδικασία του προγραμματισμού γίνεται εύκολα μέσω ειδικού λογισμικού στον υπολογιστή μας και η φόρτωση των προγραμμάτων σε αυτές γίνεται μέσω διασύνδεσης USB. Στο Σχήμα 4.8 απεικονίζεται το λογισμικό μέσα από το οποίο προγραμματίζουμε μία εφαρμογή στον υπολογιστή μας και η πλακέτα Arduino UNO στην οποία πρόκειται να αποθηκεύσουμε το πρόγραμμα που γράφουμε μέσω διασύνδεσης USB.

Σχήμα 4.7: Arduino Uno και περιφερειακά για την ανάπτυξη εφαρμογών.



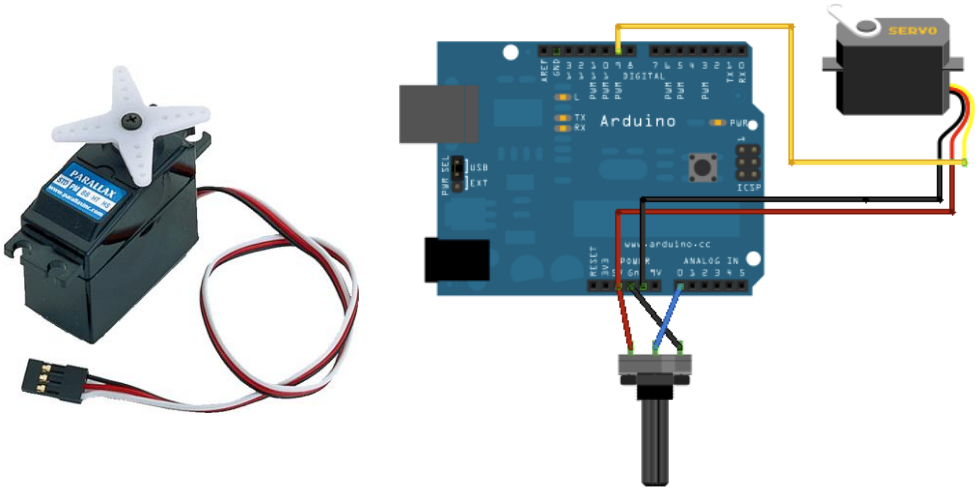
Σχήμα 4.8: Περιβάλλον του λογισμικού IDE.

Ο προγραμματισμός των πλακετών βασίζεται σε μία γλώσσα προγραμματισμού η οποία έχει σχεδιαστεί κατάλληλα για αυτές τις πλακέτες και ο προγραμματιστής μέσα από έτοιμα παραδείγματα και ένα σύνολο βιβλιοθηκών που βρίσκονται ενσωματωμένες μέσα στο περιβάλλον προγραμματισμού, μπορεί να δώσει σε μία πλακέτα οποιαδήποτε λειτουργία επιθυμεί. Μπορεί να οδηγήσει τις εισόδους και τις εξόδους της πλακέτας καθώς και να διασυνδέσει και να οδηγήσει κατάλληλα μία ή περισσότερες εξωτερικές συσκευές με ευκολία (π.χ. οθόνη LCD). Με αυτόν τον τρόπο μπορούμε εύκολα να υλοποιήσουμε εφαρμογές από την οδήγηση ενός λαμπτήρα πατώντας ένα κουμπί έως και την αυτόματη οδήγηση ενός μικρού ρομπότ.



4.9.1 Οδήγηση σερβοκινητήρων με το Arduino

Κάνοντας χρήση της βιβλιοθήκης Servo που συμπεριλαμβάνεται μέσα στο λογισμικό Arduino IDE και υλοποιώντας την ακόλουθη συνδεσμολογία με μία πλακέτα Arduino, μπορούμε να οδηγήσουμε με ευκολία ένα σερβοκινητήρα (Σχ. 4.9).



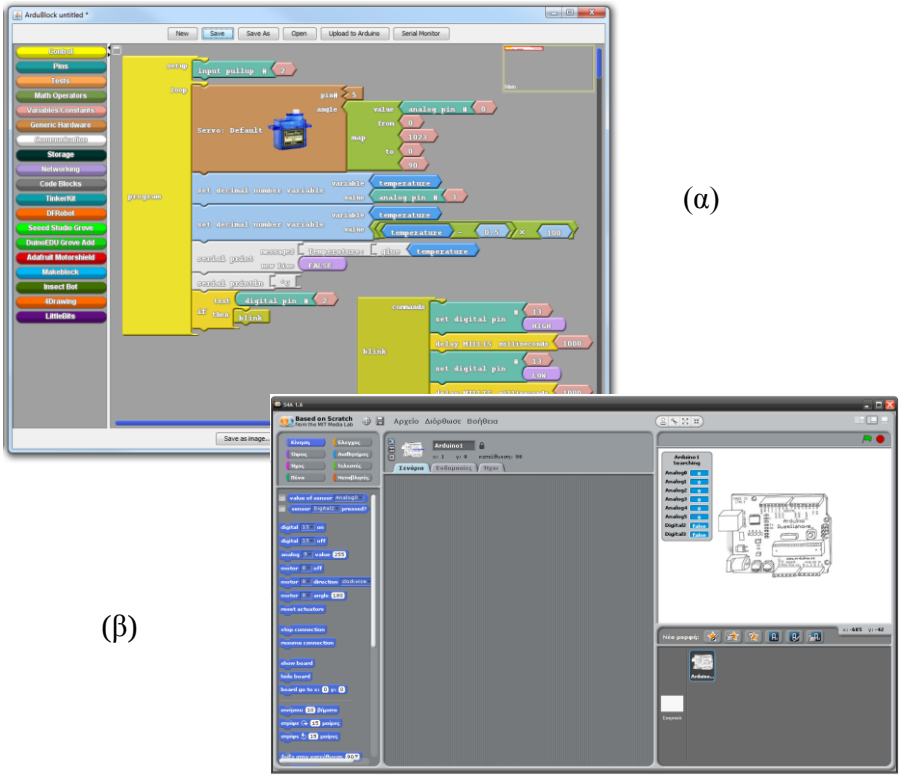
Σχήμα 4.9: Συνδεσμολογία σερβοκινητήρα με την πλατφόρμα Arduino.

Στη συνέχεια παρουσιάζεται ο κώδικας από το παράδειγμα knob.ino του περιβάλλοντος του Arduino File →Examples → Servo → Knob**.**

```
#include <Servo.h>
Servo myservo;  // Διαδικασία ελέγχου σέρβο.
int potpin = 0;
int val;

void setup(){
 myservo.attach(9);  // Έλεγχος σέρβο από ακροδέκτη 9.
}

void loop() {
val = analogRead(potpin);       //   Ανάγνωση ποτενσιόμετρου από 0 έως1023.
val = map(val, 0, 1023, 0, 180);     // Ορισμός εισόδου και γωνία περιστροφής.
 myservo.write(val);                  // Εγγραφή για κίνηση του σέρβο.
 delay(15);
```

4.9.2 Προγραμματισμός Arduino με πλακίδια

Μέσω του προγράμματος ArduBlock (http://blog.ardublock.com/) (http://sourceforge.net/projects/ardublock/) μπορούμε να προγραμματίσουμε ένα Arduino με χρήση πλακιδίων όπως απεικονίζεται στο Σχήμα 4.10α, ενώ με το εργαλεία S4A (http://s4a.cat) (Σχ. 4.10β) το όποιο είναι βασισμένο στον προγραμματισμό του Scratch μπορούμε να διασυνδέσουμε το Arduino για ανάπτυξη απλών εφαρμογών στις οποίες χειριζόμαστε το Arduino ως κάρτα συλλογής δεδομένων σε μόνη σύνδεση του Arduino με τον υπολογιστή για πολύ απλές εφαρμογές.



Σχήμα 4.10: Περιβάλλον λογισμικών α) ArduBlock και β) S4A.

Μπορείτε να ξεκινήσετε την εμπειρία σας με το Arduino από την παρακάτω επίσημη διεύθυνση:

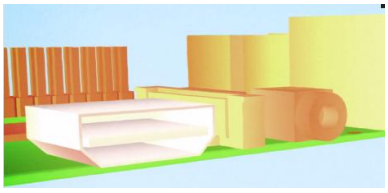
http://www.arduino.cc/

4.10 Η πλατφόρμα Raspberry Pi

Το RaspberryPi είναι μία εκπαιδευτική πλατφόρμα που υλοποιήθηκε από την εταιρία Raspberry Pi Foundation με στόχο να προωθήσει την επιστήμη των υπολογιστών στα σχολεία και στα εκπαιδευτικά ιδρύματα. Με μία πολύ μικρού μεγέθους πλακέτα η οποία περιέχει όλα τα απαραίτητα λειτουργικά στοιχεία που απαιτούνται για τη δόμηση ενός ηλεκτρονικού υπολογιστή, το Raspberry Pi μπορεί να συνδεθεί εύκολα με οθόνη, πληκτρολόγιο και ποντίκι και αμέσως δίνει την δυνατότητα ενασχόλησης με την πλατφόρμα αυτή.

4.10.1 Συνδέοντας το Raspberry Pi

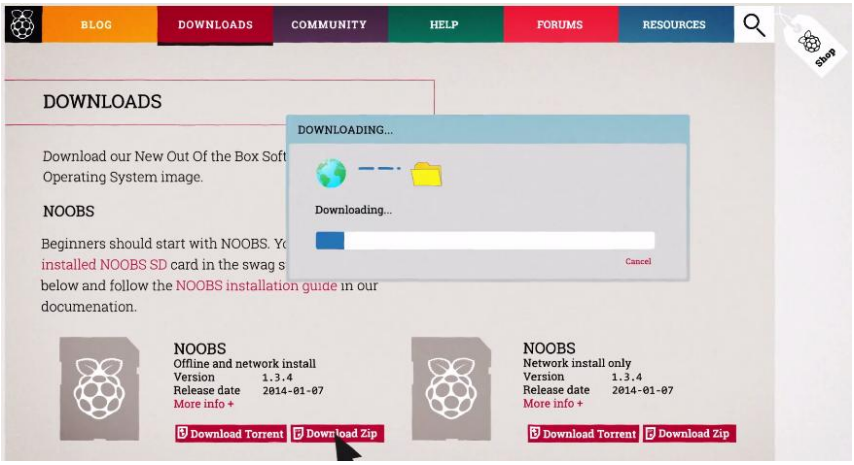
Η διασύνδεση με την οθόνη πραγματοποιείται με καλώδιο HDMI μέσω της θύρα HDMI που βρίσκεται ενσωματωμένη πάνω στην πλακέτα Raspberry Pi.



Χρησιμοποιώντας τις θύρες USB που βρίσκονται ενσωματωμένες πάνω στην πλακέτα μπορούμε εύκολα να συνδέσουμε ένα πληκτρολόγιο και ένα ποντίκι.

Μέσω ενός τροφοδοτικού MicroUSB μπορούμε να τροφοδοτήσουμε την πλακέτα. Χρησιμοποιώντας μία κάρτα SD αποθηκεύουμε το λειτουργικό σύστημα Raspberry και τοποθετούμε την κάρτα στην υποδοχή SD που διαθέτει η πλακέτα Raspberry Pi. Το λειτουργικό σύστημα του Raspberry Pi μπορούμε να το προμηθευτούμε από την κεντρική ιστοσελίδα στην ακόλουθη διεύθυνση (Σχ. 4.11).

https://www.raspberrypi.org/downloads/



Σχήμα 4.11: Λήψη του λειτουργικού NOOBS.

Υπάρχουν πολλά λειτουργικά συστήματα που μπορούμε να επιλέξουμε από την ιστοσελίδα. Για να κάνει κανείς πιο εύκολη την πρώτη του επαφή με την πλατφόρμα Raspberry Pi καλό θα είναι να επιλέξει το λειτουργικό σύστημα NOOBS στο οποίο υπάρχουν εύκολα βήματα για την εγκατάσταση του λειτουργικού συστήματος. Μέσα στο NOOBS περιέχεται και επιλογή για την εγκατάσταση του λειτουργικού συστήματος Raspbian.

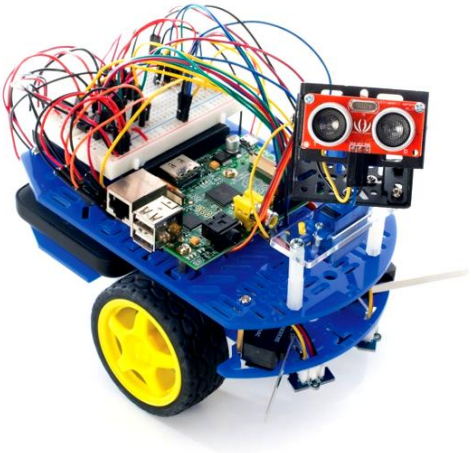
Το Raspbian είναι ένα ελεύθερο λειτουργικό σύστημα που βασίζεται στην διανομή Debian του Linux το οποίο έχει ρυθμιστεί κατάλληλα για το υλικό (hardware) του Raspberry Pi. Το λειτουργικό Raspbian έχει περίπου 35.000 προ εγκατεστημένα πακέτα (packages) και βρίσκεται ακόμα σε διαδικασία υλοποίησης προσπαθώντας να βελτιώσει την σταθερότητά του και την απόδοσή του. Συνδέοντας ένα καλώδιο Ethernet στην υποδοχή RJ-45 του Raspberry Pi μπορούμε να συνδεθούμε στο διαδίκτυο και να χρησιμοποιήσουμε το Raspberry Pi ως προσωπικό υπολογιστή.

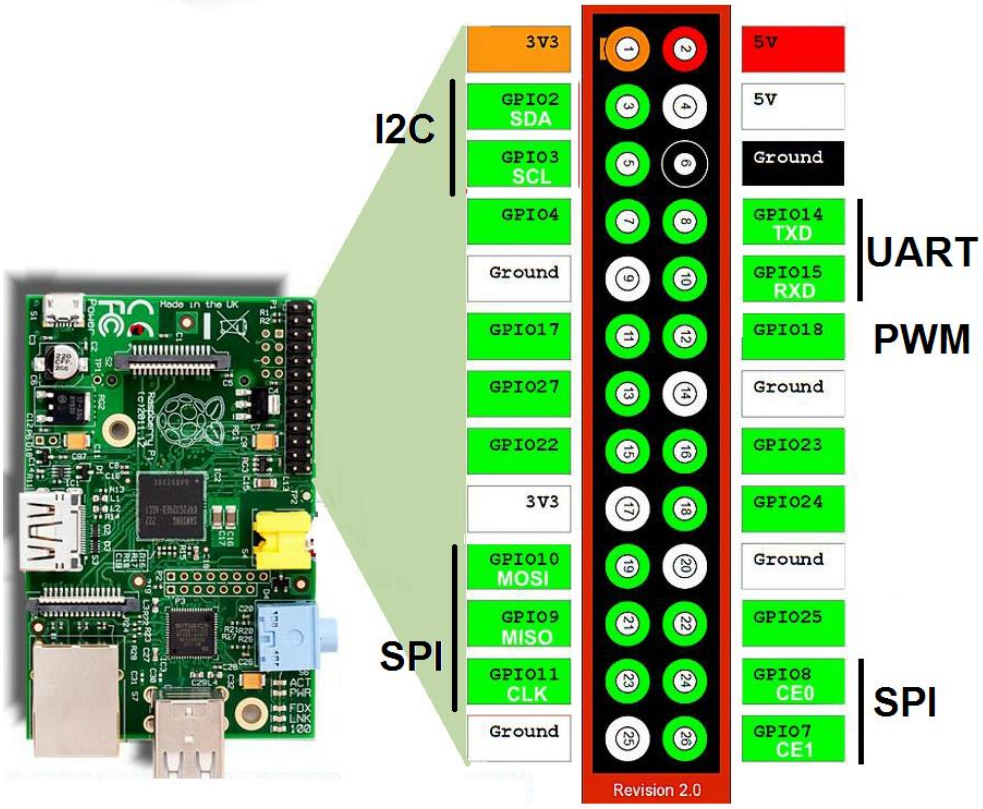
4.10.2 Χρήση του Raspberry Pi ως ενσωματωμένο σύστημα

Συγκριτικά με έναν ηλεκτρονικό υπολογιστή το Raspberry Pi μας δίνει μία επιπλέον δυνατότητα να οδηγούμε ένα σύνολο ψηφιακών εισόδων και εξόδων γενικού σκοπού GPIO (GeneralPurposeInputOutput).

Αυτές τις ψηφιακές εισόδους και εξόδους μπορούμε να τις χρησιμοποιήσουμε για να διασυνδέσουμε εξωτερικές περιφερειακές μονάδες (modules) οι οποίες εκτελούν διάφορες λειτουργίες όπως οδήγηση συσκευών ελέγχου, επικοινωνία με αισθητήρες για μέτρηση φυσικών μεγεθών (θερμοκρασία, φωτεινότητα, κ.α.) οδήγηση μηχανισμών ρομποτικών συσκευών, οδήγηση οθονών LCD και TFT κ.α.

Στο Σχήμα 4.12 απεικονίζεται το λειτουργικό διάγραμμα των εισόδων\εξόδων του Raspberry Pi.

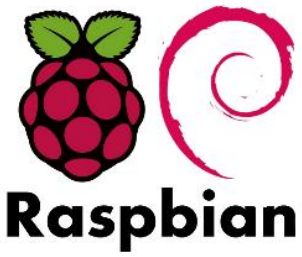


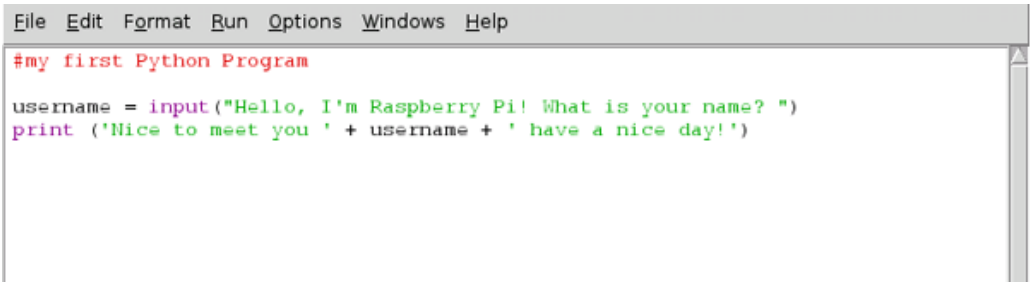


Σχήμα 4.12: Οι ακροδέκτες GPIO του Raspberry Pi.

4.10.3 Προγραμματισμός με το Raspberry Pi

Στην περίπτωση που έχουμε εγκαταστήσει στο Raspberry Pi το λειτουργικό σύστημα Raspbian, υπάρχει προ εγκατεστημένη μέσα σε αυτό η γλώσσα προγραμματισμού Python καις συγκεκριμένα υπάρχει το περιβάλλον προγραμματισμού που καλείται IDLE3. Στο Σχήμα 4.13 παρουσιάζεται ένα απλό πρόγραμμα HelloWorld σε γλώσσα Python στο περιβάλλον IDLE.





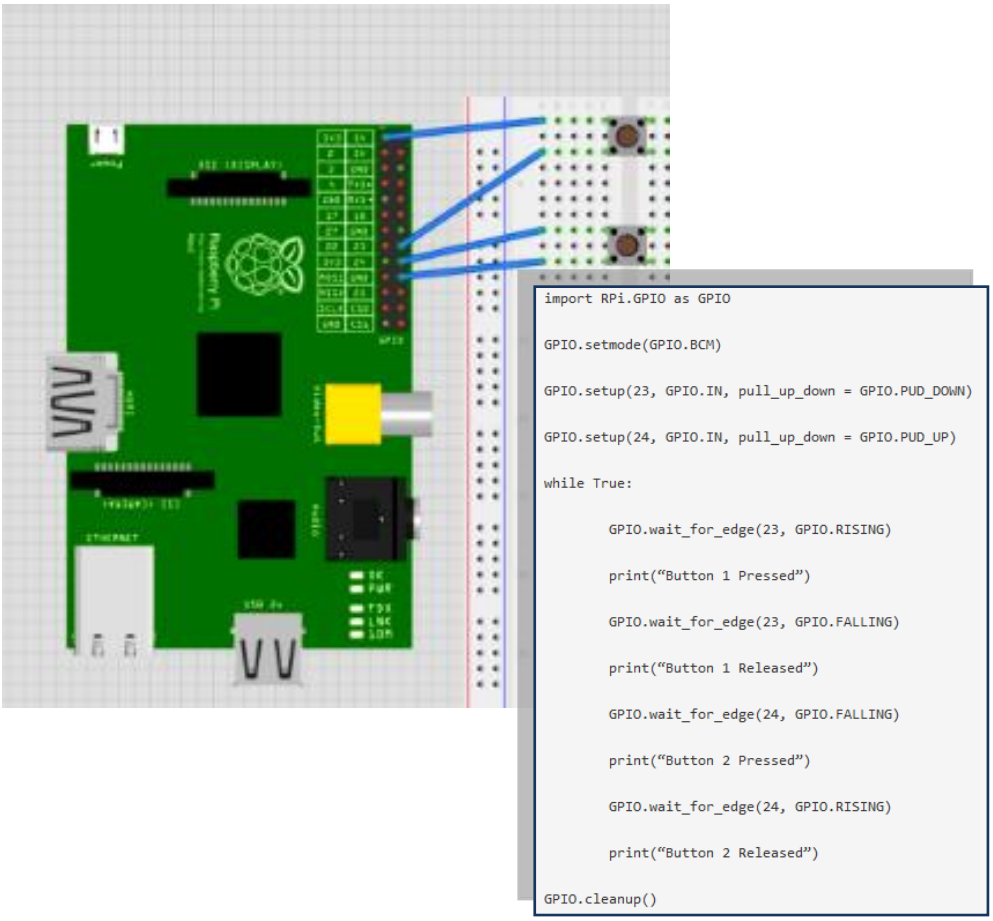
Σχήμα 4.13: Πρόγραμμα σε Python στο περιβάλλον IDLE του Raspberry Pi.

Αποθηκεύοντας το παραπάνω πρόγραμμα και επιλέγοντας Run από το κεντρικό μενού το πρόγραμμα θα μεταγλωττιστεί αυτόματα και θα εκτελεστεί ρωτώντας τον χρήση *Hello, I'm Raspberry Pi! What is your name?* Σε αυτό το σημείο περιμένει από τον χρήστη να εισάγει το όνομά του. Αφού ο χρήστης πληκτρολογήσει το όνομά (π.χ. Kostas) του εμφανίζεται το μήνυμα *Nice to meet you Kostas have a nice day!*

Με παρόμοιο τρόπο μπορούμε να γράψουμε οποιοδήποτε πρόγραμμα επιθυμούμε χρησιμοποιώντας τις εντολές της γλώσσας προγραμματισμού Python.

4.10.4 Οδήγηση GPIO του Raspberry Pi με την Python

Χρησιμοποιώντας τις βιβλιοθήκες που βρίσκονται εγκατεστημένες στην Python για την οδήγηση των εισόδων/εξόδων γενικού σκοπού της πλακέτας Raspberry Pi μπορούμε εύκολα να γράψουμε προγράμματα σε γλώσσα Python που οδηγούν κατάλληλα αυτές τις εισόδους/εξόδους. Φτάνει μόνο να συμπεριλάβουμε την κατάλληλη βιβλιοθήκη GPIO στον κώδικά μας και έπειτα να χρησιμοποιήσουμε τις συναρτήσεις και τις διαδικασίες για να υλοποιήσουμε εφαρμογές (Σχ. 4.14).



```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

GPIO.setup(23, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)

GPIO.setup(24, GPIO.IN, pull_up_down = GPIO.PUD_UP)

while True:

        GPIO.wait_for_edge(23, GPIO.RISING)

        print("Button 1 Pressed")

        GPIO.wait_for_edge(23, GPIO.FALLING)

        print("Button 1 Released")

        GPIO.wait_for_edge(24, GPIO.FALLING)

        print("Button 2 Pressed")

        GPIO.wait_for_edge(24, GPIO.RISING)

        print("Button 2 Released")

GPIO.cleanup()
```

Σχήμα 4.14: Οδήγηση GPIO του Raspberry Pi με την Python.

Μπορείτε να ξεκινήσετε την εμπειρία σας με το Raspberry Pi από την παρακάτω επίσημη διεύθυνση:

https://www.raspberrypi.org/

4.11 Ο ρομποτικός μηχανισμός Codie

Το Codie είναι ένας ξύλινος ρομποτικός μηχανισμός ο οποίος φέρει ένα σύνολο από αισθητήρες (Σχ. 4.15). Ο προγραμματισμός του γίνετε μέσω εφαρμογής για iOS. Με τον ρομποτικό μηχανισμό ο χρήστης μαθαίνει βασικό προγραμματισμό μέσα από τις κινήσεις του Codie.

Σχήμα 4.15: Ο ξύλινος ρομποτικός μηχανισμός Codie και προγραμματισμός μέσω κινητού τηλεφώνου.

Μπορείτε να ξεκινήσετε την εμπειρία σας με το Raspberry Pi από την παρακάτω επίσημη διεύθυνση:

http://getcodie.com/

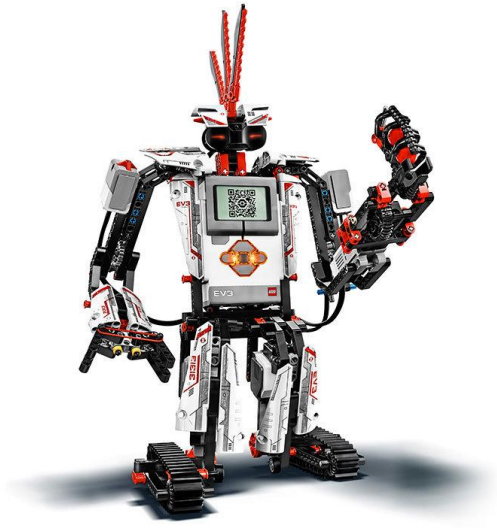
4.12 Ο ρομποτικός μηχανισμός LEGO Mindstorms EV3 και NXT

Το LEGO Mindstorms EV3 και το LEGO Mindstorms NXT είναι ένας ρομποτικός μηχανισμός ο οποίος φέρει ένα σύνολο από αισθητήρες και μπορεί με χρήση LEGO να πάρει οποιαδήποτε μορφή (Σχ. 4.16). Ο προγραμματισμός του γίνετε από μέσω του λογισμικού Mindstorms Edu NXT το οποίο ανήκει στην κατηγορία του οπτικού προγραμματισμού. Μπορεί επίσης να προγραμματιστεί και σε άλλες γλώσσες όπως ς προγραμματίζεται με όλες σχεδόν τις γνωστές γλώσσες προγραμματισμού όπως C, C++, Java, .Net, κ.α..

Μπορείτε να ξεκινήσετε την εμπειρία σας με το LEGO Mindstorms EV3 από την παρακάτω επίσημη διεύθυνση:

http://www.lego.com/en-us/mindstorms

(α)



(α) (γ)

Σχήμα 4.16: α) Υλικά LEGO, β) Mindstorms EV3 και γ) Mindstorms NXT.

Βιβλιογραφία

[1] *Ανάπτυξη εφαρμογών σε προγραμματιστικό περιβάλλον*, 2005. Π.Ι.

[2] Αντώνιος Γκοτσίνας, Κωνσταντίνος Καλοβρέκτης, 2013. *Πληροφοριακά συστήματα οικονομικών και διοικητικών επιστημών*, Εκδόσεις Βαρβαρήγου, ISBN 978-960-7996-53-4.

[3] Κωνσταντίνος Καλοβρέκτης, 2012. *Βασικές δομές ενσωματωμένων συστημάτων*, Εκδόσεις Βαρβαρήγου, ISBN 978-960-7996-48-0.

[4] *Τεχνολογία υπολογιστικών συστημάτων και λειτουργικά συστήματα,* 2005. Π.Ι.

[5] Hennessy J. L., Patterson D. A., 2003. *Αρχιτεκτονική υπολογιστών,* Εκδόσεις Τζιόλα, ISBN 978-960-418-326-5

Πηγές διαδικτύου

http://appinventor.mit.edu/explore/

http://appstudio.windows.com/en-us

http://getcodie.com/

http://hourofcode.com/gr

http://smallbasic.com/

http://www.arduino.cc/

http://www.getbusy.gr/

http://www.getcoding.gr/

http://www.kodugamelab.com/

http://www.lego.com/en-us/mindstorms

https://scratch.mit.edu/

https://www.dreamspark.com/

https://www.raspberrypi.org/

https://www.touchdevelop.com/

Γνωρίζεις πως κάθε αλληλεπίδραση ανάμεσα σε άνθρωπο και υπολογιστή γίνεται μέσα από τον κώδικα;

Είτε χρησιμοποιείς τον υπολογιστή, την ταμπλέτα, το κινητό, ή οποιαδήποτε άλλη ηλεκτρονική συσκευή ο κώδικας είναι εκεί! Μπορεί να μην τον βλέπεις, αλλά ο κώδικας είναι εκεί. Βρίσκεται παντού και είναι απαραίτητος στην καθημερινή επαφή με την τεχνολογία.

HEPIS
HELLENIC PROFESSIONALS INFORMATICS SOCIETY
ΕΛΛΗΝΙΚΟ ΔΙΚΤΥΟ ΕΠΑΓΓΕΛΜΑΤΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Γιατί να μάθω κώδικα;

Σε λίγα χρόνια όσοι δεν έχουν κάποιες βασικές δεξιότητες προγραμματισμού θα θεωρούνται 'αναλφάβητοι'!

Σε αυτό συμφωνούν οι ειδικοί, αλλά και οι 'γίγαντες' της πληροφορικής όπως ο Bill Gates, και ο Steve Jobs. Άνθρωποι που άλλαξαν τη ροή της ιστορίας, παρά το μεταξύ τους ανταγωνισμό, ενώθηκαν για να περάσουν στους νέους ένα μήνυμα:

'Επενδύστε στην Πληροφορική και τον προγραμματισμό, γιατί σου μαθαίνει να σκέφτεσαι'.

Πηγή: http://www.getcoding.gr/

</getcoding>

ISBN 978-960-93-6991-6