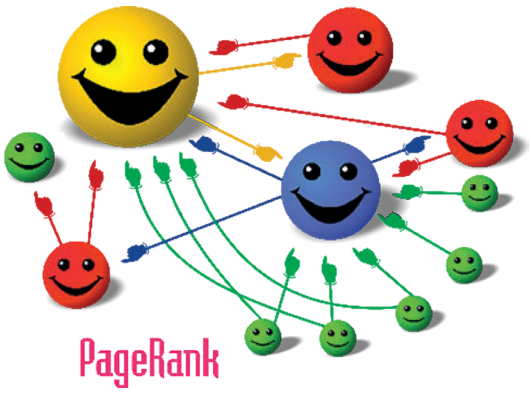
ΕΝΟΤΗΤΑ 8 **ΑΛΓΟΡΙΘΜΙΚΗ**

Ανάλυση Προβλήματος ▪ Η Έννοια του Αλγορίθμου ▪ Γλώσσες Προγραμματισμού

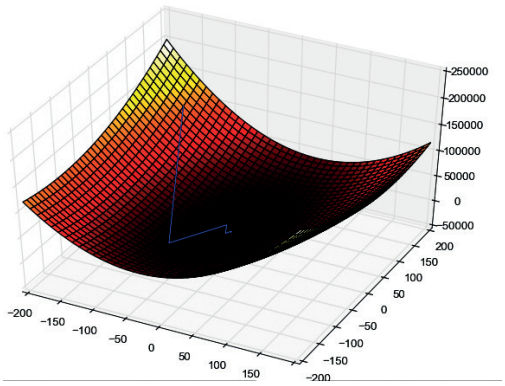
8.1 Εισαγωγή

Έχετε αναρωτηθεί ποτέ πώς μια μηχανή αναζήτησης κατατάσσει τα αποτελέσματα που επιστρέφει; Πώς το YouTube σάς προτείνει συγκεκριμένα βίντεο, ή το Netflix συγκεκριμένες ταινίες; Πώς ο πλοηγός στο κινητό σας υπολογίζει τη συντομότερη διαδρομή από το σπίτι στο σχολείο; Όλες αυτές οι ερωτήσεις έχουν μια κοινή απάντηση: τον αλγόριθμο. Φανταστείτε τον αλγόριθμο σαν μια ακολουθία από βήματα για να φτάσετε στη λύση ενός προβλήματος. Η μελέτη των αλγορίθμων, γνωστή ως **Αλγοριθμική**, αποτελεί το θεμέλιο της επιστήμης της Πληροφορικής. Η επιστήμη της Πληροφορικής, αν και συνδέεται άμεσα με την ανάπτυξη των υπολογιστικών συστημάτων, έχει τις ρίζες της βαθιά στην αρχαιότητα. Πριν την εμφάνιση των ηλεκτρονικών υπολογιστών, μεγάλοι μαθηματικοί και φιλόσοφοι είχαν ήδη αναπτύξει αλγορίθμους που παραμένουν θεμελιώδεις στην επιστήμη των υπολογιστών. Ένα από τα πιο χαρακτηριστικά παραδείγματα είναι ο αλγόριθμος του Ευκλείδη για τον υπολογισμό του μέγιστου κοινού διαιρέτη (ΜΚΔ), που χρονολογείται περίπου στο 300 π.Χ. και εξακολουθεί να χρησιμοποιείται ευρέως λόγω της απλότητάς του και της αποδοτικότητάς του. Ένα άλλο σημαντικό παράδειγμα είναι το κόσκινο του Ερατοσθένη, που αναπτύχθηκε γύρω στο 200 π.Χ. και παρέχει έναν αποτελεσματικό τρόπο για την εύρεση όλων των πρώτων αριθμών μέχρι έναν δεδομένο αριθμό. Αυτοί τεκμηριώνουν ότι η αλγοριθμική σκέψη, που είναι η κεντρική έννοια της Πληροφορικής, αποτελεί αναπόσπαστο μέρος της ανθρώπινης διανόησης και προϋπήρχε πολύ πριν την εμφάνιση των πρώτων υπολογιστών.

Σ' αυτή την ενότητα θα εξερευνήσουμε τις βασικές έννοιες και αρχές που διέπουν την αλγοριθμική σκέψη. Θα ξεκινήσουμε με τον ορισμό του αλγορίθμου και θα προχωρήσουμε στη μελέτη των χαρακτηριστικών που τον καθιστούν αποδοτικό και χρήσιμο. Ένα από αυτά είναι ο τρόπος αναπαράστασής του. Για αυτόν τον λόγο επιλέγουμε μια γλώσσα προγραμματισμού. Σε αυτήν την ενότητα, θα δείτε πολλά παραδείγματα σύγχρονων γλωσσών προγραμματισμού.



Εικόνα 8.1 Ο αλγόριθμος PageRank με βάση τον οποίο κατατάσσει τις ιστοσελίδες η μηχανή αναζήτησης Google



Εικόνα 8.2 Ο αλγόριθμος Gradient Descent με τον οποίο μαθαίνουν τα νευρωνικά δίκτυα

	2	3	4	5	6	7	8	9	10	2	3	5	7
11	12	13	14	15	16	17	18	19	20	11	13	17	19
21	22	23	24	25	26	27	28	29	30	23	29		
31	32	33	34	35	36	37	38	39	40	31	37		
41	42	43	44	45	46	47	48	49	50	41	43	47	
51	52	53	54	55	56	57	58	59	60	53	59		
61	62	63	64	65	66	67	68	69	70	61	67		
71	72	73	74	75	76	77	78	79	80	71	73	79	
81	82	83	84	85	86	87	88	89	90	83	89		
91	92	93	94	95	96	97	98	99	100	97			
101	102	103	104	105	106	107	108	109	110	101	103	107	109
111	112	113	114	115	116	117	118	119	120	113			

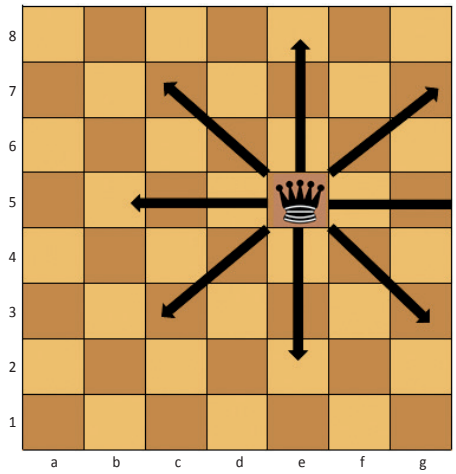
Εικόνα 8.3 Το κόσκινο του Ερατοσθένη για τον υπολογισμό των πρώτων αριθμών μέχρι το 120

8.2 Ανάλυση προβλήματος

Καθημερινά ερχόμαστε αντιμέτωποι με μια σειρά προβλημάτων διαφορετικής δομής και δυσκολίας. Υπάρχουν τα προβλήματα που καλούμαστε να λύσουμε στο πλαίσιο των μαθημάτων μας, όπως μια άσκηση στη Φυσική ή στα Μαθηματικά ή μια άσκηση γραμματικής στα Αρχαία Ελληνικά. Υπάρχουν, όμως, και τα προβλήματα που συναντάμε στην καθημερινότητά μας, όπως η εύρεση της συντομότερης διαδρομής από το σπίτι στο σχολείο, η οργάνωση του δωματίου μας ή η στελέχωση της ομάδας μπάσκετ του σχολείου. Εκτός από τα προβλήματα που αντιμετωπίζουμε στην καθημερινή μας ζωή, υπάρχουν και τα μεγάλα προβλήματα που αντιμετωπίζει η ανθρωπότητα, όπως το ενεργειακό πρόβλημα, οι πανδημίες ιών, οι κοινωνικές ανισότητες και πολλά άλλα. Τι εννοούμε όμως με την λέξη πρόβλημα;

Πρόβλημα είναι μια κατάσταση η οποία απαιτεί λύση που δεν είναι γνωστή.

Μια κατηγορία δύσκολων προβλημάτων είναι τα προβλήματα συνδυαστικής, τα οποία πρέπει να βρουν τον σωστό συνδυασμό που ικανοποιεί κάποια κριτήρια. Ένα πολύ γνωστό πρόβλημα αυτής της κατηγορίας, που μπορείτε να λύσετε με χαρτί και μολύβι, είναι το πρόβλημα των 8 βασιλισσών. Σύμφωνα με αυτό το πρόβλημα, πρέπει να τοποθετήσετε σε μια σκακιέρα 8 βασίλισσες χωρίς να απειλούνται μεταξύ τους. Δηλαδή, να μη βρεθούν δύο βασίλισσες στην ίδια γραμμή, στήλη ή διαγώνιο.



Δοκιμάστε να λύσετε το πρόβλημα στο τετράδιό σας. Στη συνέχεια, να συγκρίνετε τη λύση που βρήκατε με αυτές των συμμαθητών και συμμαθητριών σας. Τι παρατηρείτε;

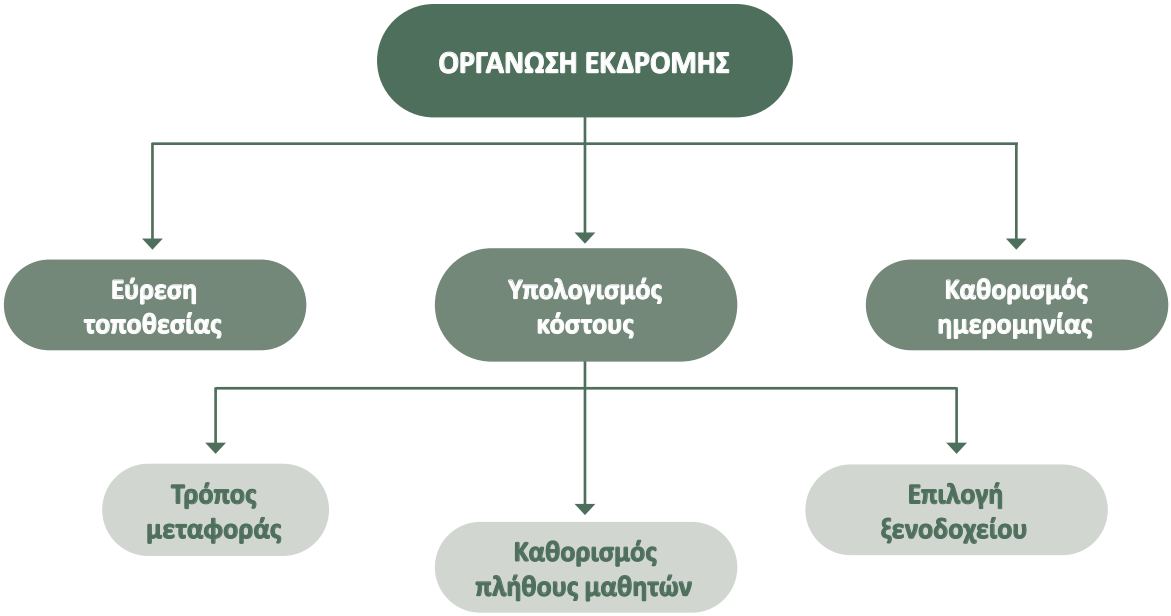
Ας εξετάσουμε πρώτα ένα απλό και ευχάριστο πρόβλημα, όπως αυτό της οργάνωσης μιας σχολικής εκδρομής:

Για να επιλύσουμε ένα πρόβλημα ξεκινάμε από την καταγραφή της ανάλυσης του προβλήματος και των βημάτων που απαιτούνται για την επίλυσή του.

Αρχικά, θέτουμε κάποια ερωτήματα στα οποία θα πρέπει να δοθούν απαντήσεις που είναι απαραίτητες για τον σχεδιασμό της εκδρομής, όπως:

- Πότε θα γίνει η εκδρομή;
- Πού θα πάμε;
- Πώς θα πάμε;
- Πόσο θα κοστίσει;

Για να απαντηθεί το τελευταίο ερώτημα, πρέπει πρώτα να απαντηθούν και άλλα ερωτήματα, όπως πόσοι μαθητές και μαθήτριες θα εκδηλώσουν ενδιαφέρον για την εκδρομή, σε ποιο ξενοδοχείο θα διαμείνουν και για πόσες μέρες. Αυτό είναι το στάδιο της **κατανόησης του προβλήματος**. Αφού απαντήσουμε στα παραπάνω ερωτήματα, θα πρέπει να καταστρώσουμε ένα σχέδιο. Θα πρέπει να θέσουμε τις ενέργειες που πρέπει να γίνουν σε μια σειρά. Σε αυτές τις περιπτώσεις, μπορεί να βοηθήσει ένας πίνακας ή ένα επεξηγηματικό διάγραμμα. Αυτό είναι το στάδιο του **σχεδιασμού της λύσης του προβλήματος**.



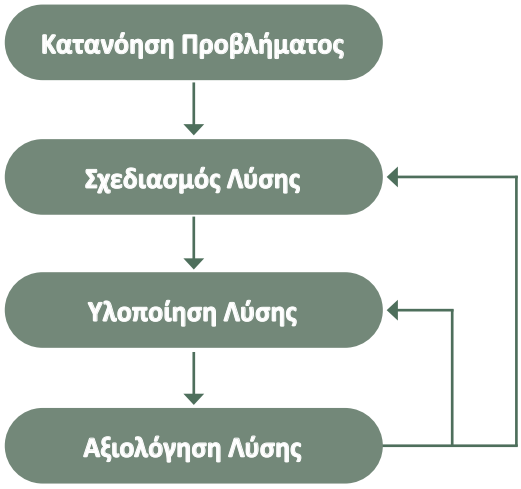
Εικόνα 8.4 Διαγραμματική αναπαράσταση της οργάνωσης μιας σχολικής εκδρομής

Ακολουθεί η υλοποίηση του σχεδίου της λύσης του προβλήματος. Κατά την υλοποίηση, υπάρχει πιθανότητα να εμφανιστούν διάφορα προβλήματα, όπως, για παράδειγμα, κάποιοι μαθητές και μαθήτριες να αλλάξουν γνώμη για την εκδρομή ή να κλείσει το ξενοδοχείο που έχουμε επιλέξει. Η επίλυση αυτών των προβλημάτων απαιτεί επανασχεδιασμό της λύσης.

Ένα από τα πιο γνωστά μοντέλα επίλυσης προβλήματος είναι το μοντέλο του Polya:

Σύμφωνα με το μοντέλο αυτό η διαδικασία της ανάλυσης και επίλυσης ενός προβλήματος μπορεί να δομηθεί σε τέσσερα βήματα:

1) Κατανόηση προβλήματος που περιλαμβάνει την καταγραφή των δεδομένων και των ζητούμενων.
2) Επινόηση – Δημιουργία ενός σχεδίου.
3) Εκτέλεση – Υλοποίηση του σχεδίου.
4) Αξιολόγηση της λύσης – Ανασκόπηση.



Στο τελευταίο στάδιο, γίνεται αναθεώρηση της αποτελεσματικότητας και της αποδοτικότητας της στρατηγικής επίλυσης που ακολουθήθηκε με σκοπό τη βελτίωση της λύσης. Για παράδειγμα, μπορεί να βρεθεί συντομότερος δρόμος για τη μετάβαση από το ξενοδοχείο στο γήπεδο ποδοσφαίρου, λόγω αυξημένης κίνησης στο αρχικό δρομολόγιο.

Παράδειγμα 1

Ένας βαρκάρης θέλει να περάσει ένα πρόβατο, έναν λύκο και ένα καφάσι με λάχανο στην απέναντι όχθη ενός ποταμού. Η βάρκα όμως είναι μικρή και μπορεί να μεταφέρει, εκτός από τον ίδιο, άλλο ένα από τα ζώα ή το καφάσι. Αν, όμως, μείνει ο λύκος μόνος του με το πρόβατο, πιθανόν να το φάει. Επίσης, το πρόβατο μπορεί να φάει το λάχανο, αν δεν υπάρχει ένας άνθρωπος να το σταματήσει.

Μπορείτε να δώσετε οδηγίες στον βαρκάρη για το πώς πρέπει να κάνει τη μεταφορά τους;

Απάντηση

Αν πάρουμε τον λύκο, τότε το πρόβατο θα φάει το λάχανο, ενώ αν πάρουμε το λάχανο τότε ο λύκος θα φάει το πρόβατο. Η μόνη μας επιλογή είναι, στην αρχή να μεταφέρουμε το πρόβατο αφού ο λύκος μπορεί να μείνει μόνος του με το λάχανο. Τα πρώτα βήματα της λύσης του προβλήματος φαίνονται παρακάτω:

Βάλε το πρόβατο στη βάρκα

Πήγαινε στην απέναντι όχθη

Άφησε το πρόβατο στην όχθη

Πήγαινε στην αρχική όχθη

Βάλε τον λύκο στη βάρκα

Πήγαινε στην απέναντι όχθη

Άφησε τον λύκο στην όχθη

Βάλε το πρόβατο στη βάρκα

Μπορείτε να συμπληρώσετε τα παραπάνω βήματα, έτσι ώστε να οδηγούν στη λύση του προβλήματος;

Μπορείτε να πειραματιστείτε με την προσομοίωση του προβλήματος στην παρακάτω εφαρμογή στο φωτόδεντρο: http://photodentro.edu.gr/v/item/ds/8521/760.

Μελετώντας τη λύση του παραπάνω προβλήματος παρατηρούμε τα εξής δύο βασικά σημεία:

1) Η λύση του προβλήματος είναι μια ακολουθία από βήματα-οδηγίες.
2) Τα βήματα περιγράφονται με δύο διαφορετικούς τρόπους, δύο διαφορετικές αναπαραστάσεις. Και στις δύο περιπτώσεις υπάρχει ένα **σύνολο επιτρεπτών εντολών**. Το σύνολο αυτό περιλαμβάνει τρεις βασικές εντολές: {Άφησε, Πήγαινε, Βάλε}.

Παράδειγμα 2

Η Ηλέκτρα έχει στη διάθεσή της ένα μπιτόνι που χωράει ακριβώς 5 λίτρα, ένα μπιτόνι που χωράει ακριβώς 3 λίτρα και μια βρύση με άφθονο νερό. Η Ηλέκτρα μπορεί να γεμίσει ένα μπιτόνι με νερό από τη βρύση, να μεταφέρει το νερό από ένα μπιτόνι σε ένα άλλο, ή να αδειάσει όλο το νερό από ένα μπιτόνι. Πώς θα μετρήσει ακριβώς ένα λίτρο;

Για να περιγράψουμε πιο αποτελεσματικά τη λύση του προβλήματος σε βήματα, χρησιμοποιούμε τις εξής εντολές:

Γέμισε(Ν): Γεμίζει το μπιτόνι με χωρητικότητα Ν μέχρι πάνω

Άδειασε(Ν): Αδειάζει όλο το μπιτόνι με χωρητικότητα Ν.

Μετακίνησε(Μ, Ν): Αδειάζει το μπιτόνι με χωρητικότητα Μ σε αυτό με χωρητικότητα Ν, μέχρι το μπιτόνι με χωρητικότητα Ν να γεμίσει. Για παράδειγμα, η εντολή Μετακίνησε(5,3), αν υποθέσουμε ότι το μπιτόνι των 5 λίτρων είναι γεμάτο και το μπιτόνι των 3 λίτρων έχει 1 λίτρο, θα μεταφέρει 2 λίτρα στο τρίλιτρο και θα μείνουν 3 λίτρα στο μπιτόνι των 5 λίτρων.

Εντολή	Νερό στο μπιτόνι	
	5 lt	3 lt
	0	0
Γέμισε(5)	5	0
Μετακίνησε(5,3)	2	3
Άδειασε(3)	2	0
Μετακίνησε (5,3)	0	2
Γέμισε(5)	5	2
Μετακίνησε (5,3)	4	3
Άδειασε(3)	4	0
Μετακίνησε (5,3)	1	3

Και σε αυτήν την περίπτωση, η λύση του προβλήματος είναι μια ακολουθία από αυστηρά καθορισμένα και πεπερασμένα βήματα, 8 σε αυτήν την περίπτωση. Τέτοιες λύσεις ονομάζονται **αλγόριθμοι**. Οι αλγόριθμοι περιγράφονται σε κάποια γλώσσα προγραμματισμού. Στο παράδειγμα αυτό η γλώσσα προγραμματισμού έχει μόνο τρεις εντολές: {Γέμισε, Άδειασε, Μετακίνησε}. Αυτό είναι το σύνολο εντολών της.

Δραστηριότητα 1

Μετρήστε ακριβώς 6 λίτρα αν έχετε στην διάθεσή σας ένα δοχείο των 5 λίτρων, ένα των 7 λίτρων και μια πηγή με άφθονο νερό. Μπορείτε μόνο να γεμίζετε μέχρι πάνω και να αδειάζετε εντελώς τα δοχεία όσες φορές θέλετε.

Δραστηριότητα 2

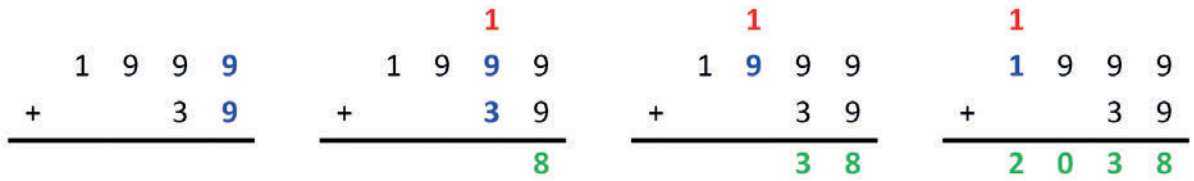
Έχετε τρεις κανάτες, μία των 10 λίτρων, μία των 7 λίτρων και μία των 3 λίτρων. Αυτή που χωράει 10 λίτρα είναι γεμάτη και οι άλλες δύο άδειες. Πώς μπορείτε να βάλετε σε μία από τις κανάτες ακριβώς 5 λίτρα νερό, χωρίς ζυγαριά, κάνοντας μόνο μεταφορές νερού από τη μία στην άλλη;

8.3 Εισαγωγή στην έννοια του αλγορίθμου

Στην προηγούμενη παράγραφο, μελετήσαμε την επίλυση προβλημάτων τα οποία επιδέχονται αλγοριθμική λύση. Δηλαδή, η λύση τους δεν είναι ένας αριθμός ή μία ποσότητα αλλά μια ακολουθία από βήματα τα οποία οδηγούν σε μια συγκεκριμένη κατάσταση. Η ακολουθία αυτή ονομάζεται αλγόριθμος.

Ο **αλγόριθμος** (**algorithm**) είναι μια ακολουθία από αυστηρά καθορισμένα βήματα που είναι εκτελέσιμα σε πεπερασμένο χρόνο και έχουν στόχο την επίλυση ενός προβλήματος.

Οι άνθρωποι χρησιμοποιούν καθημερινά διάφορους αλγορίθμους για τη διεκπεραίωση καθημερινών εργασιών. Ένας από τους πρώτους αλγορίθμους που μαθαίνουμε είναι ο αλγόριθμος της πρόσθεσης. Ένα παράδειγμα δίνεται παρακάτω:



Τα βήματα του αλγορίθμου εκτελούνται με συγκεκριμένη σειρά. Πρώτα προσθέτουμε τις μονάδες, στη συνέχεια προχωράμε στις δεκάδες κ.ο.κ.

Δραστηριότητα 3

Να καταγράψετε τα βήματα του αλγορίθμου της πρόσθεσης αναλυτικά.

Ένα άλλο κλασικό παράδειγμα αλγορίθμων αποτελούν οι συνταγές μαγειρικής. Για να μαγειρέψουμε ένα φαγητό, ακολουθούμε τις οδηγίες μιας συνταγής αυστηρά, αλλιώς το αποτέλεσμα δε θα είναι το επιθυμητό. Εδώ, εκτός από τη σειρά με την οποία θα πρέπει να εκτελεστούν τα βήματα του αλγορίθμου, σημαντικό ρόλο παίζει και η ακρίβεια με την οποία θα εκτελεστούν οι εντολές. Η παραμικρή παρέκκλιση μπορεί να προκαλέσει σοβαρά προβλήματα στο τελικό αποτέλεσμα. Για παράδειγμα, αν ρίξουμε λίγο αλάτι ή πιπέρι παραπάνω, το αποτέλεσμα μπορεί να είναι πολύ διαφορετικό από το αναμενόμενο.

Δραστηριότητα 4

Να βάλετε στη σειρά τα βήματα εκτέλεσης μιας απλής συνταγής για μακαρόνια με σάλτσα.

	Σερβίρετε τα μακαρόνια με σάλτσα ντομάτας και από πάνω ρίχνετε πιπέρι.
	Αφού στραγγίσετε τα μακαρόνια, τα βάζετε πίσω στην κατσαρόλα με λάδι και βούτυρο και ανακατεύετε.
	Βράζετε τα μακαρόνια για 8 λεπτά.
	Περιμένετε μέχρι να βράσει το νερό.
	Κλείστε το μάτι της κουζίνας.
	Ανοίξτε το μάτι της κουζίνας.
	Βάλτε τα μακαρόνια στην κατσαρόλα.
	Στραγγίστε τα μακαρόνια και στη συνέχεια σερβίρετε στα πιάτα.
	Γεμίστε την κατσαρόλα με νερό από τη βρύση.
	Τοποθετήστε την κατσαρόλα στο μάτι της κουζίνας.

Κάθε αλγόριθμος καθορίζεται από πέντε 5 χαρακτηριστικά:

- **Καθοριστικότητα:** Κάθε βήμα πρέπει να είναι αυστηρά καθορισμένο και εκτελέσιμο σε κάθε περίπτωση.
- **Περατότητα:** Ο αλγόριθμος πρέπει να τερματίζει μετά από την εκτέλεση πεπερασμένου πλήθους βημάτων.
- **Είσοδος:** Ένας αλγόριθμος μπορεί να έχει καμία, μία ή περισσότερες εισόδους, οι οποίες αντιστοιχούν στα δεδομένα που πρέπει να χρησιμοποιήσει για να επιλύσει το πρόβλημα.
- **Έξοδος:** Ένας αλγόριθμος, πρέπει να έχει τουλάχιστον μία έξοδο, η οποία αντιστοιχεί στη λύση του προβλήματος.
- **Αποτελεσματικότητα:** Κάθε βήμα του αλγορίθμου πρέπει να είναι αρκετά απλό, ώστε να μπορεί να εκτελεστεί από την υπολογιστική μηχανή για την οποία απευθύνεται. Για παράδειγμα, αν ο αλγόριθμος απευθύνεται σε ανθρώπους, θα πρέπει να μπορεί να εκτελεστεί με χαρτί και μολύβι από έναν άνθρωπο ακολουθώντας απλές και κατανοητές εντολές.

Η λέξη αλγόριθμος προέρχεται από το όνομα ενός Πέρση μαθηματικού, Abu Ja'far Mohammed ibn Musa al Khowarizmi, που έζησε τον 9ο αιώνα μ.Χ. Η λέξη «al Khowarizmi» σημαίνει «από το Khowarazm», που είναι η σημερινή πόλη Khiva του Ουζμπεκιστάν.

Ο al Khowarizmi έγραψε μια αραβική γλωσσική πραγματεία στο ινδουιστικό-αραβικό αριθμητικό σύστημα, το οποίο μεταφράστηκε στα λατινικά κατά τη διάρκεια του 12ου αιώνα. Το χειρόγραφο ξεκινά με τη φράση Dixit Algorizmi («Έτσι σπάσε τον Αλ-Κουαρίζμι»), όπου «Αλγκορίζμι» ήταν η λατινοποίηση του ονόματος του Αλ-Κουαρίζμι από τον μεταφραστή.

Μια διαδικασία, η οποία δεν πληροί το κριτήριο της περατότητας, ονομάζεται **υπολογιστική διαδικασία** (computational process). Σήμερα, σχεδόν όλες οι εφαρμογές που χρησιμοποιούμε δεν τερματίζουν ποτέ, αφού πρέπει να είναι διαθέσιμες να δεχτούν τα αιτήματα των χρηστών τους ανά πάσα στιγμή, όπως για παράδειγμα, είναι οι μηχανές αναζήτησης, οι εφαρμογές ηλεκτρονικού ταχυδρομείου, οι εφαρμογές νέφους, οι εφαρμογές Μέσων Κοινωνικής Δικτύωσης κ.λπ. Όλες αυτές οι εφαρμογές ανήκουν σε μια ειδική κατηγορία υπολογιστικών διαδικασιών γνωστών ως reactive processes, διότι αναμένουν το επόμενο αίτημα προς διεκπεραίωση από τον χρήστη. Ωστόσο, υπάρχουν και περιπτώσεις που, για κάποιον λόγο, ένα πρόγραμμα εγκλωβίζεται σε μια μη αναμενόμενη ατέρμονη διαδικασία, εξαιτίας κάποιου σφάλματος στην κωδικοποίηση του αλγορίθμου.

Αλγόριθμοι κρυπτογράφησης

Την εποχή των γαλατικών πολέμων, ο Ιούλιος Καίσαρας έγραφε στον Κικέρωνα και σε άλλους φίλους του, αντικαθιστώντας τα γράμματα του κειμένου, με γράμματα που βρίσκονται έναν αριθμό θέσεων μπροστά στο Λατινικό Αλφάβητο, σε περίπτωση που το μήνυμα έπεφτε στα χέρια του εχθρού. Ένας από τους γνωστούς και απλούς τύπους κρυπτογραφικής υποκατάστασης που χρησιμοποιούσε ήταν η μετάθεση κατά τρεις θέσεις μπροστά στο αλφάβητο.

Στην κρυπτογραφία χρησιμοποιούνται συνήθως οι όροι **κανονικό αλφάβητο** για να αναφερθούν στο αλφάβητο στο οποίο έχει γραφεί το αρχικό μήνυμα και **κρυπτογραφικό αλφάβητο** που αναφέρεται στο κρυπτογραφημένο μήνυμα. Για εύκολη κρυπτογράφηση/αποκρυπτογράφηση τοποθετούμε το κανονικό πάνω από το κρυπτογραφικό αλφάβητο, ώστε να φαίνεται πως κρυπτογραφείται κάθε γράμμα, όπως φαίνεται στο παρακάτω σχήμα.

Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	.....	Χ	Ψ	Ω
↓		↓		↓		↓		↓		↓			↓	
Χ	Ψ	Ω	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	.....	Τ	Υ	Φ

Ο αριθμός των θέσεων που μετακινείται (ολισθαίνει) το αλφάβητο είναι το μυστικό κλειδί που χρειαζόμαστε για να αποκρυπτογραφήσουμε το μήνυμα που θα λάβουμε. Πώς όμως θα κινηθούν οι κρυπταναλυτές του εχθρού, αν πέσει ένα τέτοιο μήνυμα στα χέρια τους χωρίς να ξέρουν το μυστικό κλειδί; Τι μπορεί να κάνει ο Καίσαρας για να κάνει δύσκολο το έργο τους;

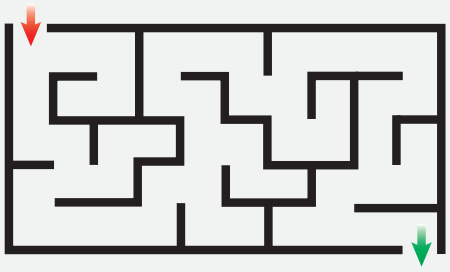
Δραστηριότητα 5. Κρυπτανάλυση

Χωριστείτε σε ομάδες των 3-4 ατόμων στην τάξη. Επιλέξτε το μυστικό κλειδί της κρυπτογράφησης και κρυπτογραφήστε με αυτό μια φράση. Στη συνέχεια, να γράψετε στον πίνακα το κρυπτογραφημένο μήνυμα. Ο στόχος είναι να βρείτε τα κλειδιά των άλλων ομάδων και να αποκρυπτογραφήσετε τα μηνύματά τους.

Να καταγράψετε αναλυτικά τη στρατηγική που θα ακολουθήσετε κατά την κρυπτογράφηση του δικού σας μηνύματος, αλλά και κατά τη διαδικασία της κρυπτανάλυσης των μηνυμάτων των άλλων ομάδων.

Πόσο χρόνο πιστεύετε ότι χρειάζεται ένας απλός υπολογιστής για να «σπάσει» των κώδικα του Καίσαρα; Υπάρχει τρόπος να τον δυσκολέψετε;

Δραστηριότητα 6. Έξοδος από Λαβύρινθο

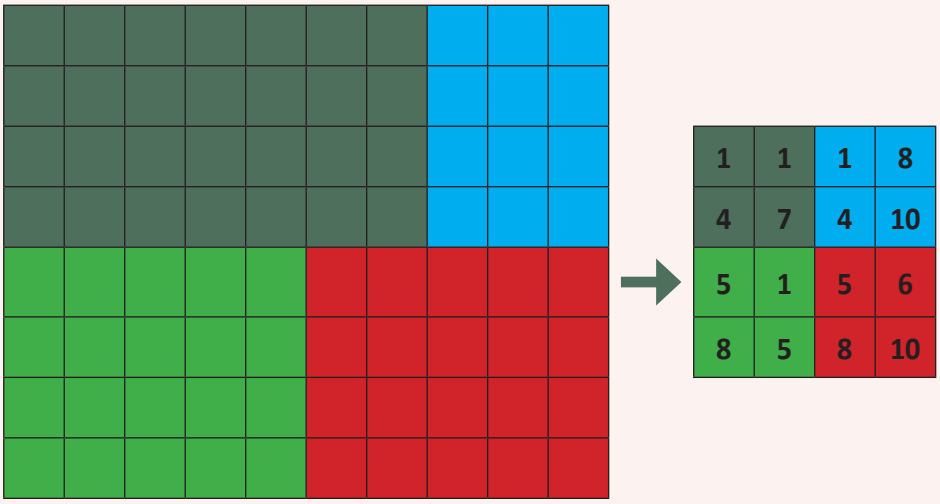


Να περιγράψετε έναν αλγόριθμο για την έξοδο από τον παρακάτω λαβύρινθο, αν υποθέσουμε ότι το κόκκινο βέλος δείχνει την είσοδο και το πράσινο βέλος δείχνει την έξοδο. Μια παλιά στρατηγική που οδηγεί, σχεδόν πάντα, στην έξοδο του λαβυρίνθου είναι ο κανόνας του δεξιού χεριού. Τοποθετούμε το δεξί μας χέρι στον τοίχο, προχωράμε και δεν το αφήνουμε ποτέ. Με αυτόν τον τρόπο, σε κάθε στροφή θα πηγαίνουμε δεξιά. Έτσι, δε θα περάσουμε ποτέ δύο φορές από το ίδιο σημείο και αν υπάρχει έξοδος θα τη βρούμε σίγουρα. Η μόνη προϋπόθεση για να λειτουργήσει αυτός ο κανόνας είναι να μην υπάρχουν κενά στον τοίχο.

Παράδειγμα 3. Αλγόριθμοι συμπίεσης δεδομένων

Στην κάρτα μνήμης του κινητού ή στον δίσκο του υπολογιστή μας αποθηκεύουμε φωτογραφίες, βίντεο, ηλεκτρονικά βιβλία και άλλες πληροφορίες. Πολλές φορές, όμως, ο αποθηκευτικός χώρος δεν είναι αρκετός. Για αυτό καταφεύγουμε στη συμπίεση μεγάλων αρχείων, έτσι ώστε να εξοικονομηθεί χώρος. Η συμπίεση δεδομένων (data compression) ορίζεται ως «η μετατροπή (μετασχηματισμός) ενός ψηφιακού αρχείου σε αρχείο μικρότερου μεγέθους με τρόπο ώστε να είναι δυνατή η αναδόμηση του συμπιεσμένου αρχείου στο αρχικό». Η διαδικασία της συμπίεσης εφαρμόζεται συστηματικά στα υπολογιστικά συστήματα που χρησιμοποιούν και επεξεργάζονται μεγάλο όγκο ψηφιακών δεδομένων. Καθημερινά χρησιμοποιούμε συμπιεσμένα αρχεία πολυμέσων, όπως είναι τα μουσικά αρχεία mp3, τα βίντεο mp4 και οι εικόνες jpeg.

Ένα πολύ απλό παράδειγμα συμπίεσης εικόνας φαίνεται στο παρακάτω σχήμα.



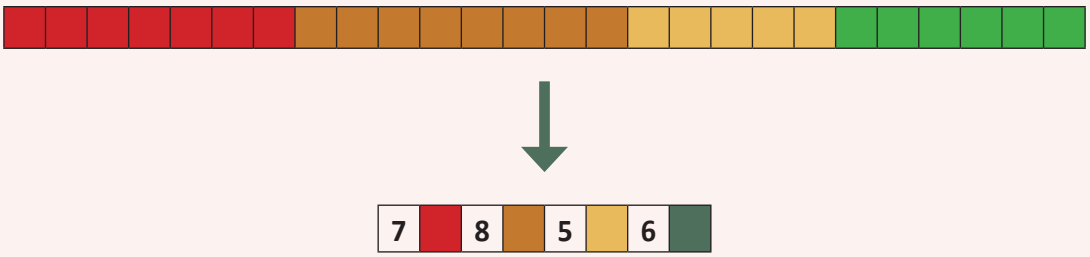
Να περιγράψετε τη βασική ιδέα πάνω στην οποία στηρίζεται ο αλγόριθμος με βάση τον οποίο έχει συμπιεστεί η παραπάνω εικόνα. Είναι αυτός ο αλγόριθμος αποτελεσματικός για όλες τις εικόνες;

Προσπαθήστε να σκεφτείτε την απάντηση πριν κοιτάξετε στην επόμενη σελίδα.

Απάντηση

Υπάρχουν πολλοί αλγόριθμοι για να καταφέρουμε να μειώσουμε το μέγεθος μίας ακολουθίας. Ένας αλγόριθμος συμπίεσης μπορεί να επιφέρει μεγάλη ελάττωση του μήκους σε ακολουθίες δεδομένων με κάποιο ιδιαίτερο χαρακτηριστικό, ενώ ο ίδιος να είναι αναποτελεσματικός για άλλες ακολουθίες, αφήνοντάς τις στο ίδιο μήκος ή ακόμα και μεγαλώνοντάς το σε μερικές περιπτώσεις.

Ο αλγόριθμος *RLE* (*Run Length Encoding*) αποτελεί έναν από τους απλούστερους αλγορίθμους συμπίεσης δεδομένων. Σύμφωνα με τη μέθοδο αυτή, διατρέχεται η ακολουθία των συμβόλων που αποτελούν τα δεδομένα προς συμπίεση και καταγράφονται οι διαδοχικές επαναλήψεις του ίδιου συμβόλου. Στη συνέχεια, αντικαθίστανται οι συνεχόμενες επαναλήψεις με το πλήθος τους, ακολουθούμενο από το σύμβολο, όπως φαίνεται στο παρακάτω παράδειγμα:



Όταν αναφερόμαστε σε σύμβολο εννοούμε αριθμό, γράμμα ή σημείο στίξης. Πολλές φορές αναφερόμαστε σε αυτό και ως χαρακτήρα.

Ο παραπάνω αλγόριθμος συμπίεσης αποδίδει αρκετά καλά, όταν έχουμε συχνές επαναλήψεις χαρακτήρων. Για παράδειγμα, η ακολουθία χαρακτήρων

AAAAAAAAAAAAA BBBBBBBBBBBBBBBBBBBBBBBBBBBBB ΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔΔ

13Α **30Β** **20Δ**

με μήκος 63 χαρακτήρες σε συμπιεσμένη μορφή χρειάζεται μόλις 9 χαρακτήρες.

Αν όμως δεν έχουμε επαναλαμβανόμενα σύμβολα, τότε με αυτήν τη μέθοδο συμπίεσης υπάρχει πιθανότητα να αυξηθεί το μέγεθος του συμπιεσμένου αρχείου, αντί να μειωθεί.

8.4 Αναπαράσταση αλγορίθμων – Γλώσσες προγραμματισμού

Η έννοια του αλγόριθμου αποτελεί το βασικό θεμέλιο της επιστήμης της Πληροφορικής και αποδεικνύει ότι η επιστήμη της Πληροφορικής υπήρχε σε λανθάνουσα μορφή πολύ πριν εμφανιστούν οι υπολογιστές. Ένας από τους πιο γνωστούς αλγορίθμους είναι ο αλγόριθμος του Ευκλείδη για τον υπολογισμό του μέγιστου κοινού διαιρέτη δύο θετικών ακέραιων αριθμών, τον οποίο περιέγραψε ο Ευκλείδης σε ένα από τα βιβλία των Στοιχείων τον 3ο αιώνα π.Χ. Θυμηθείτε ότι ο μέγιστος κοινός διαιρέτης δύο αριθμών είναι ο μεγαλύτερος ακέραιος που διαιρεί ακριβώς και τους δύο αριθμούς.

Ο Ευκλείδης σκέφτηκε έναν πολύ γρήγορο αλγόριθμο για την εύρεση του ΜΚΔ. Η βασική ιδέα είναι, ότι ο μέγιστος κοινός διαιρέτης δύο θετικών ακέραιων αριθμών **x** και **y** είναι ο ίδιος με τον μέγιστο κοινό διαιρέτη του **y** και του **x mod y**, όπου mod είναι ο τελεστής του υπολοίπου της ακέραιας διαίρεσης του **x** με τον **y**. Δηλαδή, ο αλγόριθμος βασίζεται στην απλή παρατήρηση ότι:

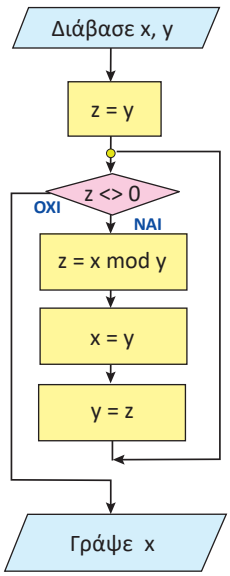
ΜΚΔ(x, y) = **ΜΚΔ**(y, x mod y)

Με διαδοχικές εφαρμογές της παραπάνω σχέσης έχουμε:

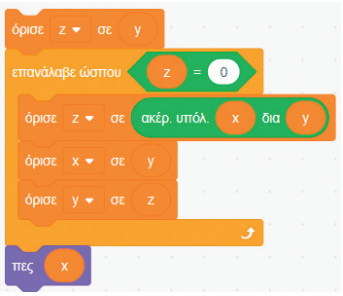
ΜΚΔ(512, 120) = ΜΚΔ(120, 32) = ΜΚΔ(32, 24) = ΜΚΔ(24, 8) = 8

x		y		υπόλοιπο	πηλίκο	
512	:	120	=	32	4	512 = 120 x 4 + 32
120	:	32	=	24	3	120 = 32 x 3 + 24
32	:	24	=	8	1	32 = 24 x 1 + 8
24	:	8	=	0	3	24 = 8 x 3 + 0

Ένας αλγόριθμος μπορεί να αναπαρασταθεί με διάφορους τρόπους. Κάποιοι από αυτούς είναι πιο κοντά στον ανθρώπινο τρόπο σκέψης και άλλοι πιο κοντά στον τρόπο λειτουργίας του υπολογιστή. Παρακάτω δίνονται τρεις εναλλακτικοί τρόποι αναπαράστασης του αλγορίθμου του Ευκλείδη σε φυσική γλώσσα, ψευδοκώδικα και διάγραμμα ροής.

Φυσική γλώσσα	Ψευδοκώδικας	Διάγραμμα Ροής
1 ΔΙΑΒΑΣΕ x, y	Διάβασε x, y	
2 ΘΕΣΕ z = y	z ← y	
3 ΑΝ z = 0 ΤΟΤΕ	Όσο z <> 0 Επανάλαβε	
ΠΗΓΑΙΝΕ ΣΤΗΝ 8		
4 ΘΕΣΕ z ← x mod y	z ← x mod y	
5 ΘΕΣΕ x = y	x ← y	
6 ΘΕΣΕ y = z	y ← z	
7 ΠΗΓΑΙΝΕ ΣΤΗΝ 3		
8 ΓΡΑΨΕ x	Τέλος_Επανάληψης	
	Γράψε x	

Ο βασικός σκοπός της σχεδίασης ενός αλγορίθμου είναι να μπορεί να εκτελεστεί από έναν υπολογιστή, ο οποίος μπορεί να εκτελέσει δισεκατομμύρια πράξεις το δευτερόλεπτο. Για να γίνει αυτό, κάθε εντολή θα πρέπει να είναι αυστηρά καθορισμένη και εκτελέσιμη από τον υπολογιστή. Δηλαδή, να αναγνωρίζει ο υπολογιστής κάθε εντολή και να ξέρει πώς θα την εκτελέσει (κριτήριο αποτελεσματικότητας). Για τον σκοπό αυτό, χρησιμοποιούμε διάφορες γλώσσες προγραμματισμού, όπως φαίνεται παρακάτω:

C++	Javascript	Python	Lua	Scratch
int z = y;	var z = y;	z = y		
while (z != 0)	while (z != 0) {	while z != 0 :	while y ~= 0 :	
{	z = x % y;	z = x % y	x , y = y, x % y	
z = x % y;	x = y;	x = y		
x = y;	y = z;	y = z	print(x)	
y = z ;	}			
}	document.write (x);	print(x)		
cout << x;				

Μεταξύ της περιγραφής ενός αλγορίθμου σε φυσική γλώσσα και της περιγραφής σε μια γλώσσα προγραμματισμού, υπάρχει μια ενδιάμεση αναπαράσταση του αλγορίθμου σε μια μορφή γνωστή ως **ψευδοκώδικας** ή **ψευδογλώσσα**, η οποία λειτουργεί ως γέφυρα μεταξύ των δύο αναπαραστάσεων, όπως φαίνεται δίπλα. Παρατηρήστε πόσο μοιάζει η αναπαράσταση του αλγορίθμου σε Scratch με την αναπαράσταση σε ψευδογλώσσα. Μπορείτε να αντιστοιχήσετε τις εντολές σε ψευδογλώσσα με αυτές σε Scratch;

```
z ← y
Όσο z ≠ 0 Επανάλαβε :
  z ← x mod y
  x ← y
  y ← z

print( x )
```

Ο ψευδοκώδικας χρησιμοποιείται όταν θέλουμε να περιγράψουμε σύνθετους και πολύπλοκους αλγορίθμους, οι οποίοι σε κάποια γλώσσα προγραμματισμού δε θα ήταν εύκολα κατανοητοί, λόγω των συντακτικών ιδιαιτεροτήτων της κάθε γλώσσας.

Grace Hopper: Μια πρωτοπόρος στον σχεδιασμό γλωσσών προγραμματισμού

Η Grace Hopper, γνωστή και ως «Amazing Grace», ήταν μια Αμερικανίδα επιστήμονας υπολογιστών και ναύαρχος του Πολεμικού Ναυτικού των Ηνωμένων Πολιτειών. Θεωρείται μια από τις πιο σημαντικές προσωπικότητες στην ιστορία της Πληροφορικής, καθώς συνέβαλε καθοριστικά στον σχεδιασμό και την ανάπτυξη της πρώτης γλώσσας προγραμματισμού υψηλού επιπέδου, της COBOL. Η Hopper θεωρείται η πρώτη που συνέλαβε την ιδέα μιας γλώσσας προγραμματισμού που θα χρησιμοποιούσε αγγλικές λέξεις αντί για τη γλώσσα μηχανής, όπου οι εντολές ήταν αριθμοί στο δυαδικό σύστημα.

Οι υπολογιστές, όμως, δεν καταλάβαιναν αγγλικό κείμενο, μόνο γλώσσα μηχανής. Έτσι, η Hopper πρότεινε την ιδέα του **μεταγλωττιστή** (**compiler**), ένα πρόγραμμα το οποίο μεταφράζει το αγγλικό κείμενο που είναι κατανοητό στους ανθρώπους σε γλώσσα μηχανής που είναι κατανοητή στους υπολογιστές. Ένας διερμηνέας, δηλαδή, μεταξύ ανθρώπου και μηχανής, κάτι που για εκείνη την εποχή φαινόταν ανέφικτο. Μάλιστα, αρκετοί συνάδελφοι της Hopper την προσγείωναν, λέγοντάς της ότι κάτι τέτοιο δε μπορεί να γίνει. Ωστόσο, η Grace Hopper επέμεινε στο όνειρό της και τα κατάφερε. Σχεδίασε τον πρώτο μεταγλωττιστή ο οποίος στη συνέχεια αξιοποιήθηκε ως βάση για την πρώτη γλώσσα προγραμματισμού υψηλού επιπέδου, την COBOL. Παρακάτω δίνεται ένα πρόγραμμα το οποίο προσθέτει όλους τους ακέραιους αριθμούς από 1 έως και 10 σε γλώσσα μηχανής σε γλώσσα Assembly και σε Python. Η Assembly (συμβολική γλώσσα) είναι μια γλώσσα που βρίσκεται ένα επίπεδο πάνω από τη γλώσσα μηχανής. Η διαφορά είναι ότι έχουν δοθεί λεκτικές περιγραφές σε κάθε ακολουθία δυαδικών ψηφίων που αναπαριστά κάποια εντολή.

Γλώσσα Μηχανής	Γλώσσα Assembly	Python
	INDEX = $01	
	SUM = $02	
10101000 00001010	LDA #10	INDEX = 10
10001100 00000001	STA INDEX	
00111100	CLA	SUM = 0
01010001 00000001	LOOP ADD INDEX	**while** INDEX > 0:
01000011 00000001	DEC INDEX	SUM += INDEX
11000000 11111010	BNE LOOP	INDEX -= 1
10001100 00000010		
11111111	STA SUM	**print**("Τελικό Άθροισμα:", SUM)
	BRK	

Σήμερα, πολλές σύγχρονες γλώσσες προγραμματισμού παράγουν μια ενδιάμεση μορφή κώδικα για μια νοητή μηχανή (virtual machine) με σκοπό την ανεξαρτησία του προγράμματος από συγκεκριμένες πλατφόρμες. Έτσι, ένα πρόγραμμα γραμμένο σε μια γλώσσα μπορεί να εκτελείται το ίδιο σε διάφορα λειτουργικά συστήματα ή υπολογιστές διαφορετικών αρχιτεκτονικών. Η μορφή αυτή κώδικα μηχανής λέγεται συνήθως **bytecode**. Παρακάτω δίνεται ένα τέτοιο παράδειγμα για τη γλώσσα Python.

Python Bytecode		Πηγαίος κώδικας σε Python
0 LOAD_CONST 2 STORE_FAST 4 LOAD_CONST 6 STORE_FAST 8 LOAD_FAST 10 LOAD_FAST 12 BINARY_ADD 14 STORE_FAST 16 LOAD_CONST 18 RETURN_VALUE	1 (28) 0 (X) 2 (496) 1 (Y) 0 (X) 1 (Y) 2 (Z) 0 (None)	def test(): X = 28 Y = 496 Z = X + Y

Όπως οι φυσικές γλώσσες, έτσι και οι γλώσσες προγραμματισμού έχουν κάποια βασικά χαρακτηριστικά, όπως:

Αλφάβητο	Είναι το σύνολο των χαρακτήρων (συμβόλων) που χρησιμοποιούνται από τη γλώσσα.	A, B, C, …, Z, a..z, 0,1..9, _
Λεξιλόγιο	Το σύνολο των λέξεων (tokens) που αναγνωρίζει η γλώσσα ως αποδεκτά και έχουν κάποια σημασία.	**this**, **while**, quantum_bit
Συντακτικό	Το σύνολο των κανόνων που ακολουθούμε για να συνδέουμε λέξεις σε αποδεκτές από τη γλώσσα εκφράσεις και εντολές.	**if** sensor == “Rainy” **then** roofTop = True

Για να συντάξουμε ένα πρόγραμμα σε κάποια γλώσσα προγραμματισμού, θα πρέπει να χρησιμοποιήσουμε ένα περιβάλλον προγραμματισμού το οποίο περιλαμβάνει μια σειρά από εργαλεία, όπως είναι ο **συντάκτης** (editor) και ο **μεταγλωττιστής** (compiler). Ο συντάκτης είναι ένα πρόγραμμα επεξεργασίας κειμένου με ειδικές όμως λειτουργίες για τη σύνταξη και τη διόρθωση του κώδικα. Ο μεταγλωττιστής είναι το πρόγραμμα το οποίο μεταφράζει ένα πρόγραμμα σε γλώσσα προγραμματισμού στη γλώσσα που είναι κατανοητή από τη μηχανή, ώστε να μπορεί να την εκτελέσει.

Το πρόγραμμα σε γλώσσα μηχανής παράγεται μόνο αν δε βρεθούν συντακτικά ή άλλα λάθη στον αρχικό πηγαίο κώδικα (source code), όπως φαίνεται δίπλα όπου η μεταβλητή d δεν έχει οριστεί.

```
double a = 1, b = 2;
int c = d;
```

CS0103: The name 'd' does not exist in the current context

Show potential fixes (Alt+Enter or Ctrl+.)

Σήμερα, αρκετοί συντάκτες κώδικα έχουν αρκετά προχωρημένες λειτουργίες, όπως η πρόβλεψη της επόμενης εντολής που έχουμε στο μυαλό μας με χρήση εργαλείων Τεχνητής Νοημοσύνης, τα οποία αξιοποιούνται πλέον από αρκετούς προγραμματιστές.

```
{
    int a = 1;
    int temp = a;
}
```

Ένα άλλο είδος μεταφραστών πηγαίου κώδικα σε γλώσσα μηχανής είναι οι **διερμηνευτές**, οι οποίοι μεταφράζουν και εκτελούν τον κώδικα εντολή-εντολή. Ενώ οι μεταγλωττιστές ελέγχουν όλο το πρόγραμμα και το μεταφράζουν σε κώδικα μηχανής μόνο αν δε βρεθούν συντακτικά λάθη, οι διερμηνευτές μπορεί να εκτελέσουν τις πρώτες εντολές και στη συνέχεια να σταματήσουν την εκτέλεση στη μέση επειδή βρέθηκε λάθος σε επόμενη εντολή.

Ένα πρόγραμμα το οποίο έχει περάσει από όλους τους ελέγχους του μεταγλωττιστή/διερμηνευτή δε σημαίνει απαραίτητα ότι έχει σωστή λειτουργία. Για παράδειγμα οι εντολές δίπλα εκτελούνται χωρίς πρόβλημα, αλλά το αποτέλεσμά τους είναι λανθασμένο. Αυτά τα λάθη λέγονται λογικά και αποτελούν τη δυσκολότερη κατηγορία λαθών στην ανίχνευση και διόρθωσή τους.

```
average = (a + b + c) / 2
```

```
if number>0 :
    print("0 αριθμός είναι αρνητικός")
```

8.5 Ερωτήσεις - Ασκήσεις

1 Διάσχιση Γέφυρας. Πέντε άνθρωποι θέλουν να περάσουν μια γέφυρα. Είναι όμως νύχτα και διαθέτουν μόνο μια λάμπα με φυτίλι διάρκειας 30 λεπτών. Η γέφυρα αντέχει το πολύ δύο άτομα, οπότε δεν μπορούν να περάσουν όλοι μαζί. Για να περάσουν τη γέφυρα χρειάζονται οπωσδήποτε τη λάμπα. Επίσης, κάθε ένας χρειάζεται διαφορετικό χρόνο για να περάσει τη γέφυρα. Ο πιο γρήγορος χρειάζεται 1 λεπτό, ο επόμενος 3 λεπτά, και υπόλοιποι 6, 8 και 12 λεπτά αντίστοιχα. Πώς θα περάσουν απέναντι;

2 Έχουμε ένα μπρίκι με νερό που βράζει και ένα αβγό που πρέπει να βράσουμε για 9 λεπτά ακριβώς. Δυστυχώς, δεν έχουμε κανένα ρολόι παρά μόνο δύο κλεψύδρες: η μία διάρκειας 7 και η άλλη 4 λεπτών. Ποιος είναι ο συντομότερος τρόπος για να μετρήσουμε 9 λεπτά;

3 Έχουμε 9 όμοια κέρματα από τα οποία το ένα είναι κάλπικο, δηλαδή πιο ελαφρύ. Επίσης, διαθέτουμε μια ζυγαριά ισορροπίας, όπως αυτή της εικόνας, με την οποία μπορούμε να συγκρίνουμε μεταξύ τους 2 βάρη.

α) Θέλουμε με 2 μόνο ζυγίσεις να εντοπίσουμε το κάλπικο κέρμα.

β) Πόσες ζυγίσεις θα χρειαστούμε για 81 κέρματα και πόσες για 36;

4 Μετρήστε ακριβώς 9 λίτρα, αν έχετε στη διάθεσή σας ένα δοχείο των 10 λίτρων, ένα των 7 λίτρων και μια πηγή με άφθονο νερό. Μπορείτε μόνο να γεμίζετε μέχρι πάνω και να αδειάζετε εντελώς τα δοχεία όσες φορές θέλετε.

5 Έχετε τρία δοχεία, ένα των 12 λίτρων, ένα των 8 και ένα των 5. Το δοχείο των 12 λίτρων είναι γεμάτο, ενώ τα άλλα δύο άδεια. Δεν έχετε άλλο νερό, μόνο τα 12 λίτρα. Να χωρίσετε την ποσότητα του νερού στη μέση, δηλαδή 6 λίτρα στο 12λιτρο και 6 λίτρα στο 8λιτρο. Μπορείτε μόνο να γεμίσετε ή να αδειάσετε τα δοχεία μέχρι τέλους, όσες φορές θέλετε. Τα δοχεία δεν έχουν καμία ογκομετρική ένδειξη επάνω τους. Σας δίνεται μια γλώσσα προγραμματισμού για το πρόβλημα των δοχείων η οποία έχει μόνο μια εντολή:

Αδειασε(Δοχείο1, Δοχείο2): Μετακινεί το περιεχόμενο του δοχείου 1 στο δοχείο 2.

π.χ. η εντολή **Αδειασε**(12, 5) όταν αρχικά το 12λιτρο είναι γεμάτο και το 5λιτρο άδειο, αδειάζει το 12λιτρο στο 5λιτρο, άρα στο 12λιτρο θα μείνουν 7 λίτρα.

6 Αν είχατε μια κανάτα των 6 λίτρων και μια των 2 λίτρων, θα μπορούσατε να μετρήσετε ακριβώς 5 λίτρα; Αιτιολογήστε την απάντησή σας.

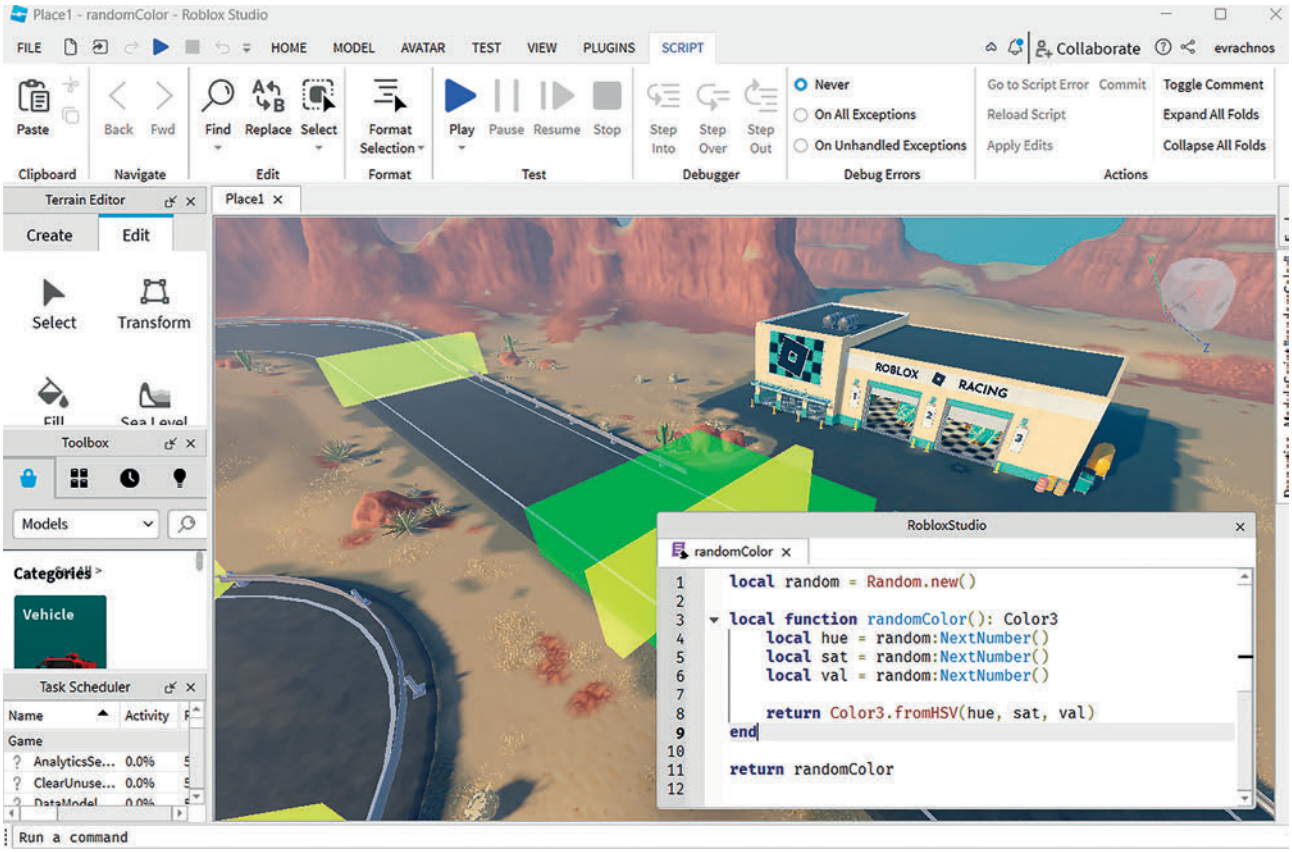
ΕΝΟΤΗΤΑ 9 **ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ**

Προγραμματισμός με το Scratch

9.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο εμβαθύναμε στην έννοια του αλγορίθμου. Οι αλγόριθμοι αποτελούν την καρδιά της επιστήμης των υπολογιστών, παρέχοντας τις απαραίτητες οδηγίες για την επίλυση προβλημάτων και την εκτέλεση υπολογιστικών εργασιών. Ωστόσο, η πραγματική δύναμη των αλγορίθμων αναδεικνύεται μόνο όταν υλοποιούνται και εκτελούνται μέσω των υπολογιστικών συστημάτων. Αυτό ακριβώς είναι το σημείο όπου οι γλώσσες προγραμματισμού παίζουν καθοριστικό ρόλο. Οι γλώσσες προγραμματισμού είναι τα εργαλεία που μας επιτρέπουν να μεταφράζουμε τους αλγορίθμους σε κατανοητές και εκτελέσιμες εντολές για τους υπολογιστές. Όλες οι εφαρμογές που χρησιμοποιούμε καθημερινά στο κινητό μας τηλέφωνο, στο τάμπλετ, στον υπολογιστή, στον εγκέφαλο του αυτοκινήτου έχουν διατυπωθεί και υλοποιηθεί σε κάποια γλώσσα προγραμματισμού.

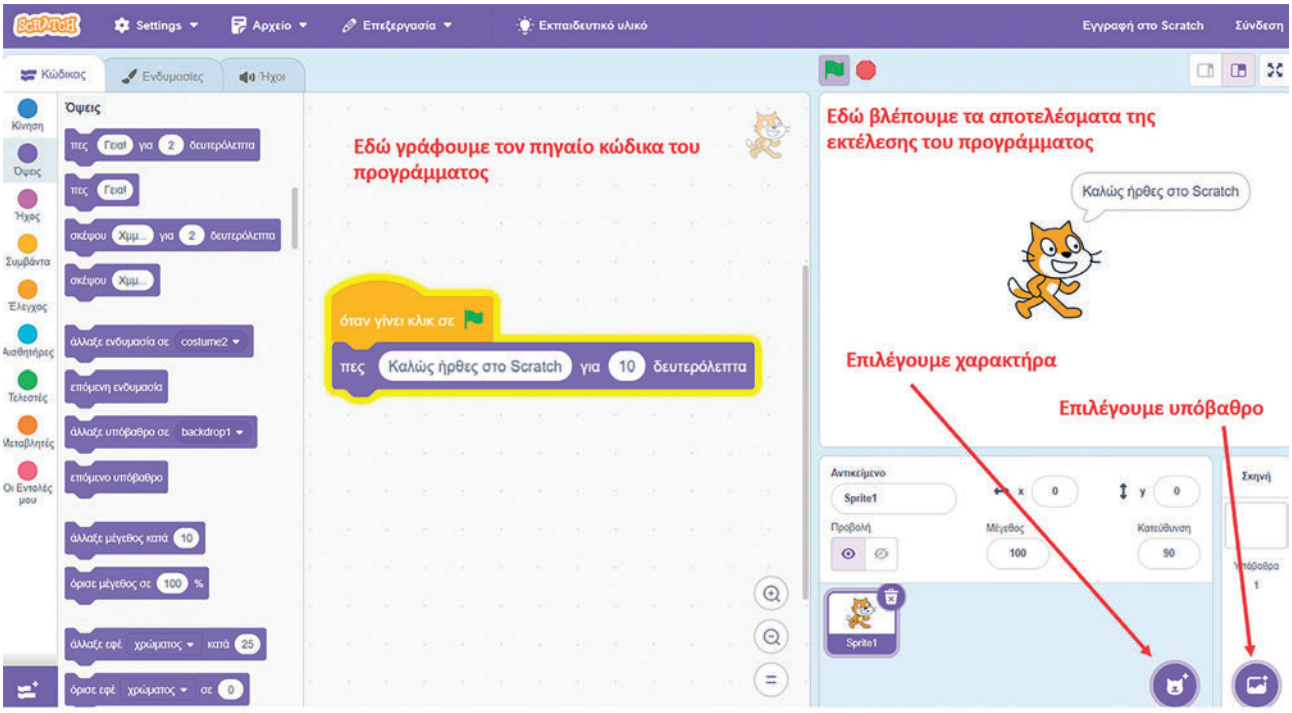
Στην ενότητα αυτή θα αναπτύξετε τα πρώτα σας προγράμματα στο περιβάλλον προγραμματισμού με πλακίδια (block-based) Scratch.



Εικόνα 9.1 Το περιβάλλον προγραμματισμού Roblox Studio IDE στο οποίο μπορούμε να αναπτύξουμε παιχνίδια για την πλατφόρμα Roblox στη γλώσσα προγραμματισμού Lua

9.2 Το περιβάλλον προγραμματισμού Scratch

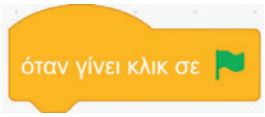
Το Scratch είναι ένα εκπαιδευτικό περιβάλλον προγραμματισμού με πλακίδια (block-based) που μας δίνει τη δυνατότητα να δημιουργούμε διαδραστικές ιστορίες, κινούμενα σχέδια, προσομοιώσεις και παιχνίδια. Όλα αυτά μπορούμε να τα μοιραστούμε μέσω του Διαδικτύου με άλλους χρήστες. Επίσης, μπορούμε να χρησιμοποιήσουμε ή να επεκτείνουμε τα προγράμματα άλλων χρηστών στην κοινότητα του Scratch. Στον δικτυακό τόπο https://scratch.mit.edu/ μπορείτε να δημιουργήσετε το δικό σας έργο ή να περιηγηθείτε στα έργα που έχουν μοιραστεί άλλοι προγραμματιστές και να επεκτείνετε κάποια από αυτά ή να δείτε τον κώδικα με τον οποίο έχουν δημιουργηθεί. Το Scratch έχει αναπτυχθεί από μια μικρή ομάδα ερευνητών του MIT.



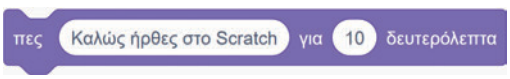
Εικόνα 9.2 Το περιβάλλον προγραμματισμού Scratch (https://scratch.mit.edu/)

Στο πλαϊνό μενού φαίνονται όλες οι θεματικές κατηγορίες εντολών, όπως είναι οι εντολές κίνησης, οι όψεις, τα συμβάντα, οι εντολές ελέγχου κ.λπ. Κάθε ομάδα εντολών έχει αντιστοιχιστεί σε ένα χρώμα. Το πρώτο μας πρόγραμμα χρησιμοποίησε δύο εντολές:

Ένα συμβάν το οποίο ξεκινάει την εκτέλεση των εντολών όταν γίνει κλικ στην πράσινη σημαία.

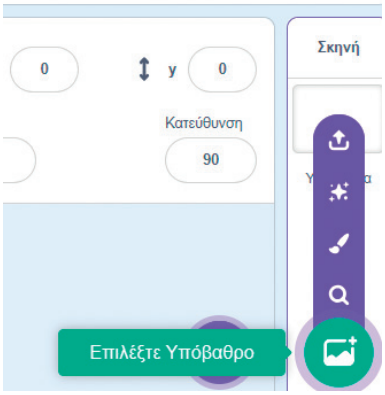


και μια εντολή από τις όψεις η οποία εμφανίζει ένα μήνυμα σε μορφή διαλόγου από τη γάτα για 10 δευτερόλεπτα.

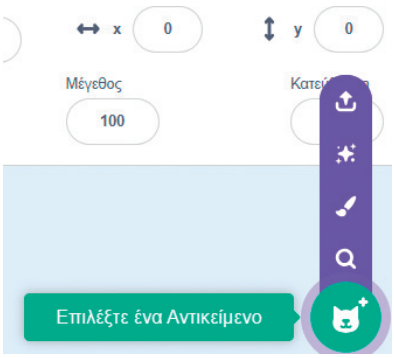


9.3 Ψηφιακή αφήγηση στο Scratch

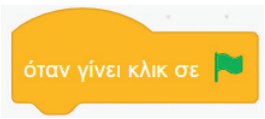
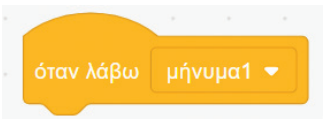
Αξιοποιώντας κάποιες από τις ομάδες εντολών των συμβάντων και των εντολών ελέγχου μπορούμε να δημιουργήσουμε μικρές διαδραστικές ιστορίες σαν σύντομα κόμικ. Κάτω δεξιά επιλέγουμε υπόβαθρο από τα ήδη προτεινόμενα ή μπορούμε να σχεδιάσουμε το δικό μας στη ζωγραφική ή να μεταφορτώσουμε (upload) μια εικόνα-υπόβαθρο που βρήκαμε στο Διαδίκτυο.



Αφού επιλέξουμε το υπόβαθρο που ταιριάζει στην ιστορία μας, πρέπει να επιλέξουμε και δύο χαρακτήρες:



Για κάθε χαρακτήρα μπορούμε να επιλέξουμε πού θα κοιτάει, αν θα κάθεται, θα περπατάει ή θα έχει κάποια άλλη στάση. Αφού επιλέξουμε τον χαρακτήρα που θέλουμε, στη συνέχεια, επιλέγουμε πάνω αριστερά στις ενδυμασίες και εκεί ορίζουμε τη στάση και την ενδυμασία του. Οι εντολές που θα χρησιμοποιήσουμε για να εμφανίζονται οι διάλογοι με τη σειρά και όχι ο ένας πάνω στον άλλον ανήκουν στην ομάδα συμβάντων.

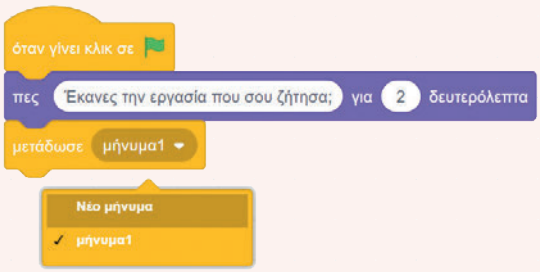
		
Όταν γίνει κλικ στην πράσινη σημαία εκτελούνται οι εντολές που ακολουθούν.	Όταν λάβω το μήνυμα από κάποιον/-α εκτελούνται οι εντολές που ακολουθούν.	Μεταδίδω το μήνυμα «μήνυμα1» το οποίο περιμένει ο άλλος χαρακτήρας.

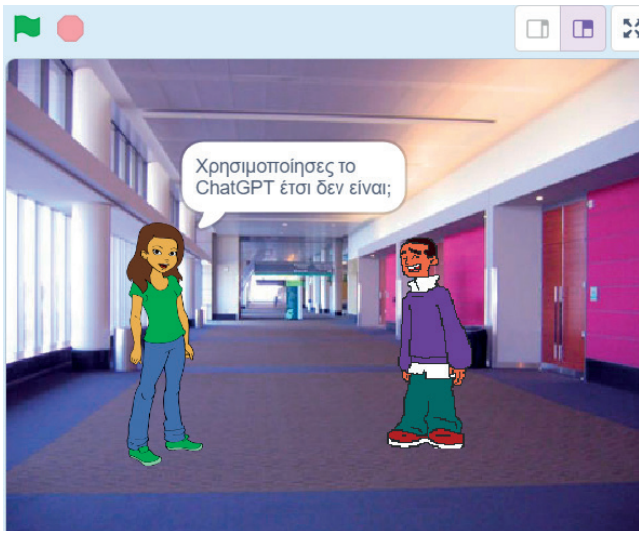
Παράδειγμα 1

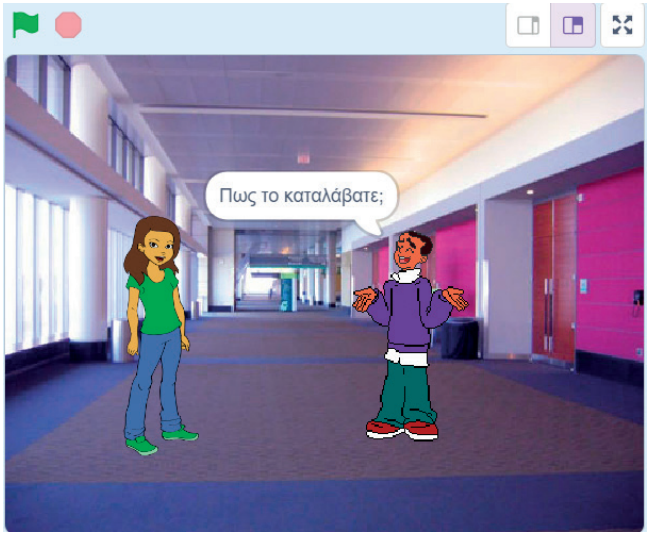
Ας επιλέξουμε δύο χαρακτήρες, την κυρία Χρύσα και τον μαθητή της Οδυσσέα.

Η κυρία Χρύσα κάνει την έναρξη της συνομιλίας με τον μαθητή. Χρησιμοποιούμε την εντολή **πες** από τις όψεις και την εντολή **μετάδωσε** μήνυμα από τα συμβάντα. Για αυτό δημιουργούμε ένα νέο μήνυμα με την εντολή <μετάδωσε **μήνυμα**> στο οποίο μπορούμε να δώσουμε ό,τι όνομα θέλουμε. Εδώ δίνουμε τον αριθμό 1. Μετά τη μετάδοση του μηνύματος η εκτέλεση του μπλοκ εντολών σταματάει.

Ο Οδυσσέας περιμένει να λάβει το μήνυμα 1 για να απαντήσει στην καθηγήτριά του και με τη σειρά του της μεταδίδει το μήνυμα 2, ώστε η καθηγήτρια να καταλάβει ότι ήρθε η σειρά της να μιλήσει. Για αυτό χρησιμοποιούμε την εντολή <όταν λάβω **μήνυμα**>. Με αυτόν τον τρόπο ο δεύτερος συνομιλητής μιλάει μόνο όταν τελειώσει η πρώτη, έτσι ώστε να μη μιλάνε ταυτόχρονα.





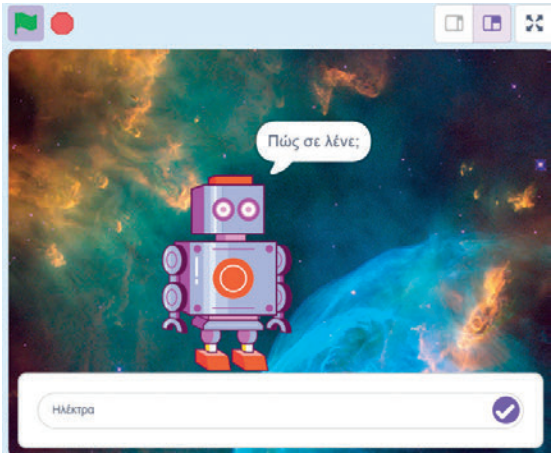


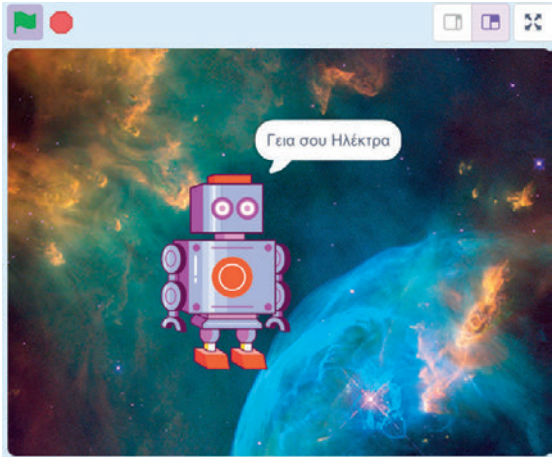
9.4 Σχεδιάζοντας ένα chatbot

Τα chatbots είναι προγράμματα για υπολογιστές τα οποία χρησιμοποιούν την Τεχνητή Νοημοσύνη και τη φυσική γλώσσα, προκειμένου να επεξεργαστούν τις ερωτήσεις κάποιου χρήστη ώστε να δώσουν αυτοματοποιημένες απαντήσεις. Η συνομιλία που εξελίσσεται δίνει πολλές φορές την εντύπωση στον χρήστη ότι έχει απέναντί του άνθρωπο και όχι μηχανή. Υπάρχουν εξελιγμένα chatbots εξειδικευμένα σε διάφορα θέματα, αλλά και απλοί βοηθοί που χρησιμοποιούμε καθημερινά μέσα από εφαρμογές στο κινητό μας.

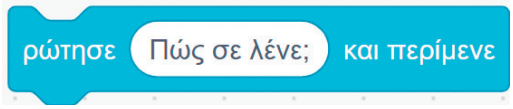
Ένα πρόσφατο παράδειγμα είναι ο ψηφιακός βοηθός του gov.gr που βασίζεται σε τεχνολογία Τεχνητής Νοημοσύνης και απαντάει σε ερωτήματα χρηστών της πύλης gov.gr.

Για να υλοποιήσουμε ένα chatbot χρειαζόμαστε εντολές που θα προσδώσουν διαδραστικότητα μέσα από ερωτήσεις που θα θέτει στον χρήστη και απαντήσεις που θα δίνει στις ερωτήσεις του χρήστη.



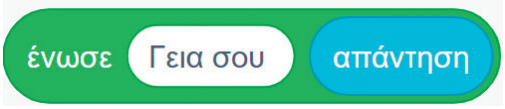


Για αυτόν τον λόγο αξιοποιούμε τις παρακάτω εντολές από τους αισθητήρες και από τους τελεστές.



Εμφανίζει ένα πλαίσιο εισόδου δεδομένων στο οποίο περιμένει από τον χρήστη να γράψει την απάντησή του.

Η απάντηση που δίνει ο χρήστης στην ερώτηση καταχωρείται σε μια περιοχή στη μνήμη που ονομάζεται «**απάντηση**».



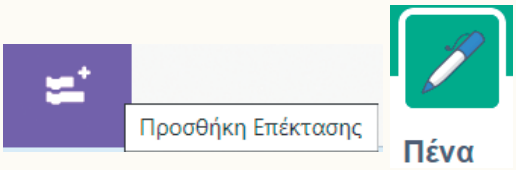
Όταν θέλουμε να ενώσουμε δύο ή περισσότερες λέξεις χρησιμοποιούμε την εντολή **ένωσε**.

Το διπλανό πρόγραμμα είναι μια πολύ απλή έκδοση ενός chatbot. Οι απαντήσεις που δίνει η μηχανή περιέχουν μέρος των απαντήσεων που δίνει ο χρήστης, ώστε να προσομοιώνεται με κάποιο τρόπο η ανθρώπινη συμπεριφορά. Μετά την εκτέλεση του προγράμματος, η θέση στη μνήμη με το όνομα **απάντηση** περιέχει την τελευταία είσοδο που δόθηκε. Κάθε φορά που η **απάντηση** δέχεται μια νέα τιμή αυτή γράφεται πάνω στην προηγούμενη η οποία έτσι διαγράφεται.

9.5 Η δύναμη της επανάληψης

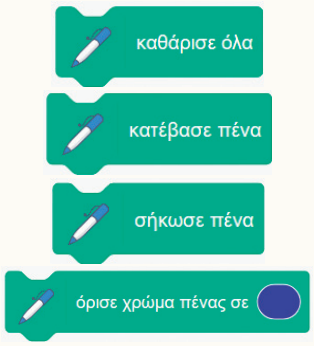
Ομάδα Εντολών με λειτουργίες πένας

Στο Scratch 3 η πένα δε βρίσκεται μέσα στις βασικές εντολές στο πλαϊνό μενού. Θα πρέπει να πάτε κάτω αριστερά στην προσθήκη επέκτασης και στη συνέχεια να επιλέξετε την πένα, ώστε να εμφανιστούν οι εντολές.

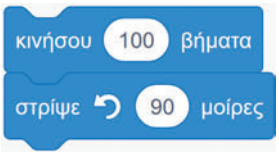


Βασικές Λειτουργίες Πένας

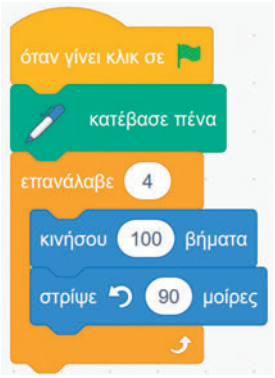
- Καθαρίζει όλη την οθόνη από όλα τα σχήματα που έχουμε σχηματίσει.
- Κατεβάζει την πένα για το αντικείμενο, έτσι ώστε να αφήνει πίσω του ένα ίχνος όταν κινείται και με αυτόν τον τρόπο να ζωγραφίζει σχήματα.
- Ανεβάζει την πένα έτσι ώστε το αντικείμενο να μην αφήνει ίχνος όταν κινείται.
- Ορίζει το χρώμα της πένας.

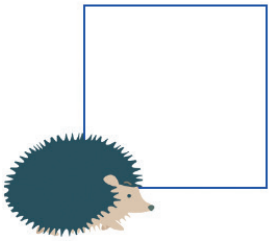


Για να σχηματίσει ένα τετράγωνο, ο σκαντζόχοιρος θα πρέπει να σχεδιάσει τέσσερις γραμμές. Όταν φτάσει στο τέλος της γραμμής, θα πρέπει να στρίψει 90° έτσι ώστε να σχηματιστεί ορθή γωνία. Παρατηρούμε ότι οι εντολές **κινήσου** και **στρίψε** επαναλαμβάνονται τέσσερις φορές.

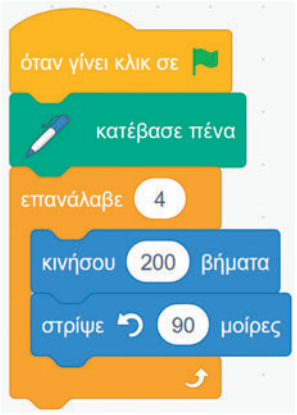


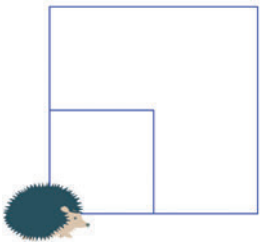
Αντί λοιπόν να τις γράφουμε μπορούμε να χρησιμοποιήσουμε την εντολή **Επανάλαβε** από την ομάδα εντολών ελέγχου. Η εντολή αυτή εκτελεί τις εντολές που είναι οριοθετημένες μέσα στην επανάληψη τόσες φορές όσες θέλουμε. Εδώ έχουμε δώσει τον αριθμό 4. Αν θέλαμε να σχεδιάσουμε ένα δεκάγωνο, θα γλιτώναμε αρκετές γραμμές κώδικα.



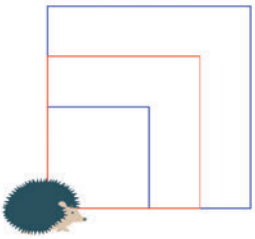


Είναι, όμως, το μόνο πλεονέκτημα της εντολής επανάληψης ότι γλιτώνουμε χώρο; Ας δούμε το διπλανό παράδειγμα, όπου αποφασίσαμε να σχεδιάσουμε ένα τετράγωνο με διπλάσια πλευρά. Παρατηρώντας το τελικό αποτέλεσμα, διαπιστώνουμε ένα πρόβλημα. Ενώ θέλαμε να σχηματιστεί ένα μόνο τετράγωνο πλευράς 200, βλέπουμε και ένα τετράγωνο πλευράς 100.





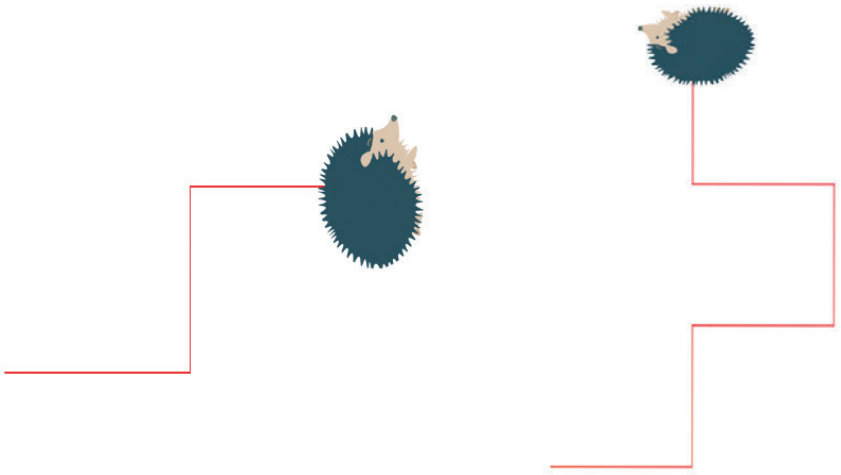
Πώς προέκυψε αυτό; Για να διερευνήσουμε περισσότερο αυτό το πρόβλημα αλλάζουμε πάλι την πλευρά του τετραγώνου και την κάνουμε 150 για να δούμε τι θα συμβεί. Ταυτόχρονα, όμως, αλλάζουμε το χρώμα της πένας σε κόκκινο. Αυτό που κάνουμε αυτή τη στιγμή λέγεται **εκσφαλμάτωση** (debugging). Ψάχνουμε, δηλαδή, να βρούμε ένα bug (σφάλμα) στον κώδικά μας. Έτσι καταλήγουμε στο διπλανό πρόγραμμα.



Έχουμε προσθέσει και μια εντολή **περίμενε** ώστε να φαίνεται η κίνηση του σκαντζόχοιρου, ενώ σχεδιάζει το σχήμα. Εκτός από την εντολή **περίμενε**, έχουμε αυξήσει και την πλευρά του τετραγώνου από 100 σε 150. Πόσες αλλαγές θα χρειάζονταν αν δε χρησιμοποιούσαμε την εντολή επανάληψης αλλά γράφαμε όλες τις εντολές όπως πριν; Αυτό αποτελεί ένα σημαντικό πλεονέκτημα όταν γράφουμε κώδικα. Να μπορούμε να πετύχουμε τον σκοπό μας με τις ελάχιστες δυνατές αλλαγές. Η δυνατότητα αυτή λέγεται **επεκτασιμότητα** (extensibility) του κώδικά μας, γιατί μας δίνει τη δυνατότητα να προσαρμόζουμε τις εφαρμογές μας εύκολα στις νέες εξελίξεις χωρίς πολλές αλλαγές.

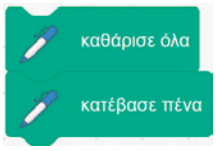
Τώρα φαίνεται τι έχει συμβεί. Το κόκκινο τετράγωνο είναι αυτό που σχηματίστηκε με την τελευταία εκτέλεση του προγράμματός μας. Τα άλλα δύο σχηματίστηκαν από τις προηγούμενες εκτελέσεις και παρέμειναν εκεί γιατί δε δώσαμε εντολή να καθαρίσει η οθόνη.

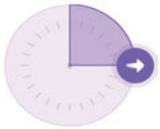
Ας δούμε και το επόμενο παράδειγμα, όπου ο σκαντζόχοιρος δε σχεδιάζει ένα κλειστό σχήμα, με αποτέλεσμα να μην επιστρέψει στη θέση από την οποία ξεκίνησε. Το δεύτερο σχήμα προκύπτει από την εκτέλεση του ίδιου προγράμματος για δεύτερη φορά. Ο σκαντζόχοιρος όχι μόνο δεν έχει επιστρέψει στο σημείο εκκίνησης, αλλά δείχνει προς την αντίθετη κατεύθυνση.



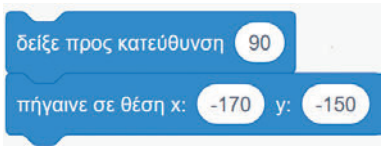
Αυτό που έπρεπε να γίνει στο ξεκίνημα λέγεται **αρχικοποίηση** (initialization). Δηλαδή, να επαναφέρουμε όλα τα αντικείμενα του προγράμματός μας στην αρχική κατάσταση.

Αυτό σημαίνει ότι θα πρέπει να καθαρίσουμε το σκηνικό και να κατεβάσουμε την πένα, ώστε ο σκαντζόχοιρος να σχεδιάζει το σχήμα που θέλουμε κατά την κίνηση.





Επίσης, θα πρέπει να μεταφέρουμε τον σκαντζόχοιρο στο αρχικό σημείο από όπου ξεκίνησε και να τον στρέψουμε προς τα δεξιά.

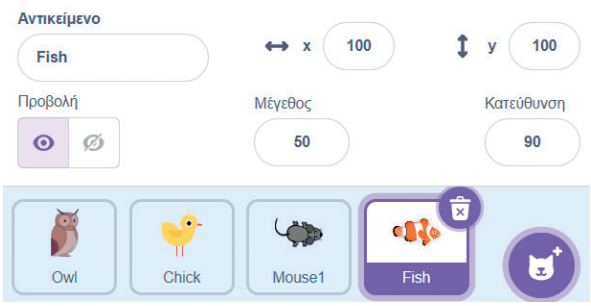


Έτσι, θα σχηματίσει γωνία 90° με τον κατακόρυφο άξονα. Οι συντεταγμένες (x, y) = (- 170, - 150) αναφέρονται στο σημείο εκκίνησης που βρίσκεται κάτω αριστερά.

Αρχικά μεταφέρουμε τον σκαντζόχοιρο στην πάνω αριστερή γωνία. Το σημείο αυτό έχει συντεταγμένες (- 170, - 150). Δηλαδή βρίσκεται 170 βήματα αριστερά από το κέντρο και 150 βήματα κάτω από το κέντρο.

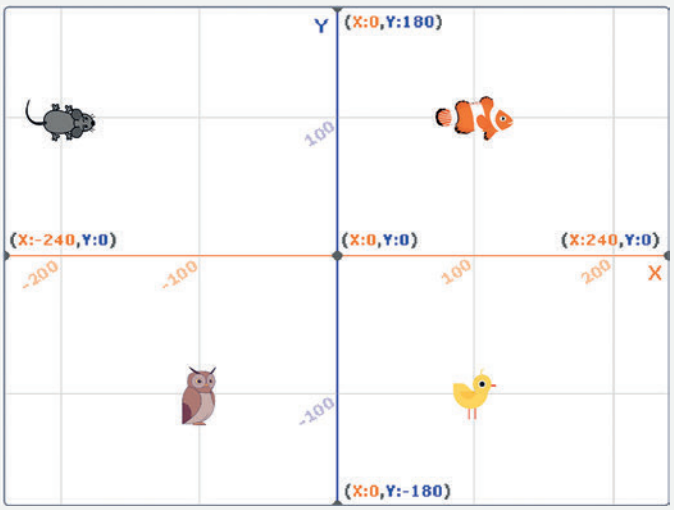
Οι συντεταγμένες ενός σημείου στο Scratch είναι η οριζόντια και κατακόρυφη απόσταση από το κέντρο της οθόνης. Άρα το κέντρο έχει συντεταγμένες (0, 0).

Αν θέλουμε να βρούμε τις ακριβείς συντεταγμένες ενός αντικειμένου αφού το μεταφέρουμε στη θέση που θέλουμε, κοιτάμε κάτω αριστερά στα χαρακτηριστικά του. Εδώ φαίνεται ότι το ψάρι του οποίου έχουμε μειώσει το μέγεθος στο μισό βρίσκεται στο σημείο με συντεταγμένες (100, 100).



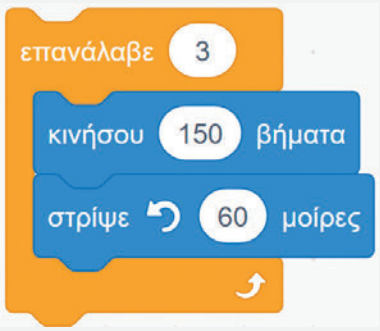
Δραστηριότητα 1

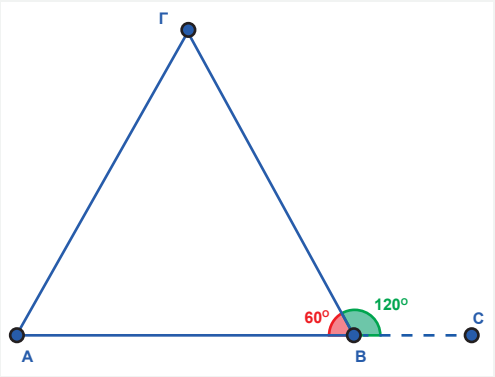
Να βρείτε τις συντεταγμένες των παρακάτω χαρακτήρων.



Δραστηριότητα 2

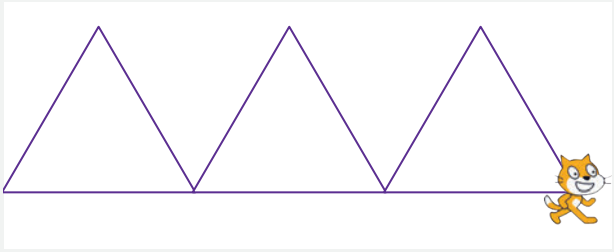
Μια μαθήτρια έδωσε τις παρακάτω εντολές για να σχεδιαστεί ένα ισόπλευρο τρίγωνο πλευράς 150. Τι σχήμα εμφανίζεται τελικά και ποιου σχήματος είναι μέρος; Τι πρέπει να διορθώσει η μαθήτρια ώστε να εμφανιστεί ισόπλευρο τρίγωνο;

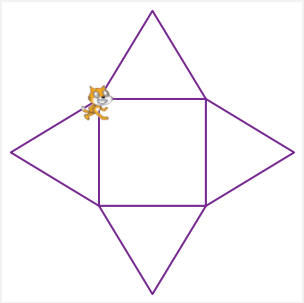


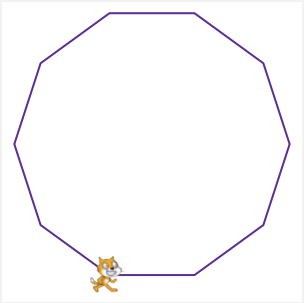


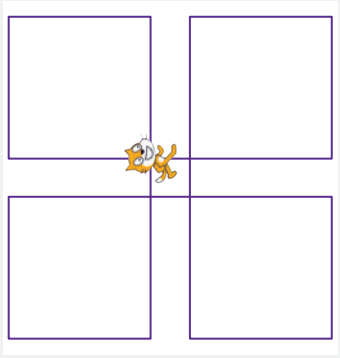
Δραστηριότητα 3

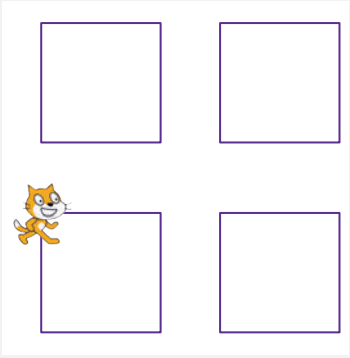
Να δώσετε τα προγράμματα που σχεδιάζουν τα παρακάτω σχήματα. Στο πρώτο σχήμα η γάτα δε μπορεί να ζωγραφίσει την ίδια γραμμή δύο φορές. Επίσης, σε κάθε σχήμα όλες οι πλευρές είναι ίσες μεταξύ τους.





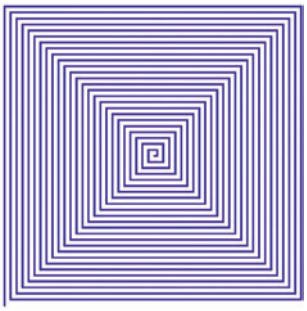




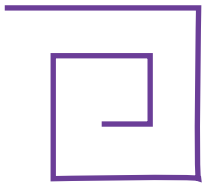


9.6 Η έννοια της μεταβλητής

Θέλουμε να σχεδιάσουμε το διπλανό σχήμα. Παρατηρούμε ότι, ενώ φαίνεται ότι κάθε φορά η στροφή είναι 90° όπως όταν σχεδιάζεται ένα τετράγωνο, το μήκος κάθε πλευράς αλλάζει συνέχεια. Θέλουμε η πλευρά να ξεκινάει από ένα αρχικό μήκος και, στη συνέχεια, να αυξάνει σταδιακά. Θα πρέπει να βρούμε έναν τρόπο να αναπαραστήσουμε την πλευρά που μεταβάλλεται μετά από κάθε κίνηση, δηλαδή μια ποσότητα που μεταβάλλεται. Γι' αυτό χρησιμοποιούμε τη **μεταβλητή**. Η μεταβλητή είναι μια θέση στη μνήμη, η οποία περιέχει μια τιμή που μπορεί να αλλάξει, όποτε θελήσουμε.



Δίπλα φαίνεται η πρώτη προσπάθεια που έκανε μια μαθήτρια. Ξεκίνησε με πλευρά 20 και σε κάθε κίνηση αύξανε την πλευρά κατά 10. Έτσι, πήρε το διπλανό σχήμα. Προφανώς, δεν μπορούμε να συνεχίσουμε έτσι, γιατί η έκταση του προγράμματος θα είναι 5-6 σελίδες.

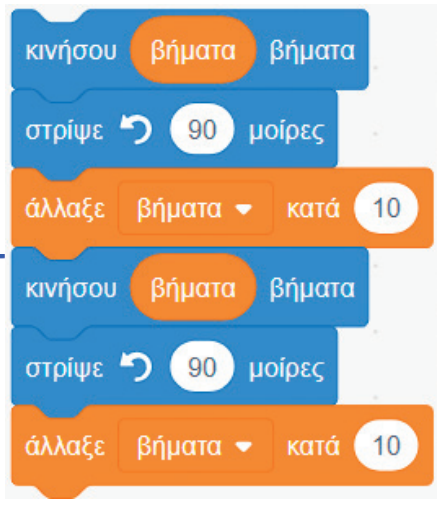


Σκεφτόμαστε, λοιπόν, πώς θα μπορούσαμε να γράψουμε αυτές τις εντολές με την μορφή επανάληψης. Θα πρέπει να βρούμε έναν τρόπο να γράψουμε έναν αλγόριθμο της μορφής:

Επανάλαβε 7 φορές
  Κινήσου βήματα
  Στρίψε αριστερά 90°
  Αύξησε την **πλευρά** κατά 10

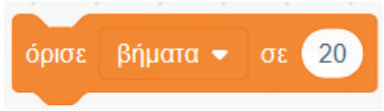
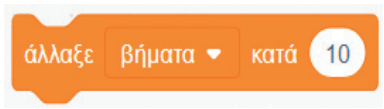
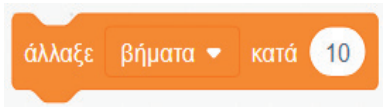
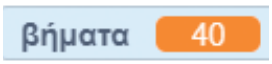
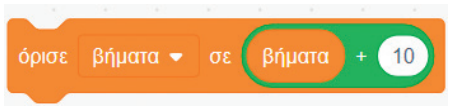
Για αυτό θα χρειαστεί να ορίσουμε μια μεταβλητή με όνομα **βήματα**.

Μετά την εισαγωγή της μεταβλητής, μπορούμε να γράψουμε το διπλανό τμήμα κώδικα με τη χρήση εντολής επανάληψης, έτσι ώστε να επιτελεί την ίδια ακριβώς λειτουργία.



Σκεφτείτε πόσες αλλαγές θα χρειαστούν σε κάθε περίπτωση αν θελήσουμε να γίνει πιο πυκνό το σπιράλ και αλλάξουμε το 10 σε 5.

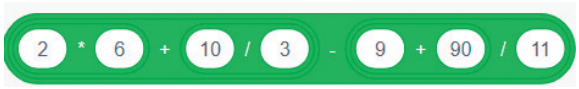
Κάθε μεταβλητή έχει ένα όνομα και αναφέρεται σε μια θέση στη μνήμη του υπολογιστή. Όταν θέσουμε μια τιμή σε αυτή τη θέση, η προηγούμενη τιμή που είχε διαγράφεται, όπως φαίνεται στο παρακάτω παράδειγμα:

Εντολή	Αποτέλεσμα στη μνήμη	Επεξήγηση εντολής
		Θέτει στη μεταβλητή βήματα την τιμή 20 βήματα ← 20
		Αυξάνει τη μεταβλητή βήματα κατά 10 βήματα ← βήματα + 10
		Αυξάνει τη μεταβλητή βήματα κατά 10 βήματα ← βήματα + 10
		Αυξάνει τη μεταβλητή βήματα κατά 10 βήματα ← βήματα + 10

Από τα παραπάνω φαίνεται ότι οι δύο τελευταίες εντολές είναι ισοδύναμες, δηλαδή εκτελούν την ίδια ακριβώς λειτουργία, αυξάνουν τη μεταβλητή κατά 10.

Εκτός από την πρόσθεση, υπάρχουν και άλλοι τελεστές που μπορούμε να χρησιμοποιήσουμε, αν χρειαστεί να κάνουμε αριθμητικούς υπολογισμούς. Επίσης, μπορούν να υπολογιστούν πιο σύνθετες αριθμητικές παραστάσεις, όπως για παράδειγμα:

$$2 \cdot 6 + \frac{10}{3} - (9 + 90) / 11$$



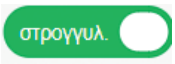
Πρόσθεση

Αφαίρεση

Πολλαπλασιασμός

Διαίρεση

Υπόλοιπο Ακέραιας Διαίρεσης

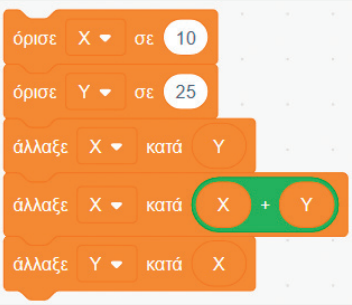


Στρογγυλοποίηση

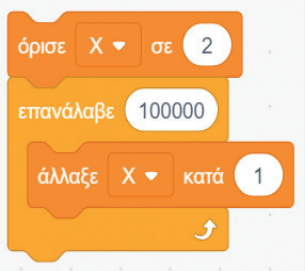
Δραστηριότητα 4

Ποιες είναι οι τελικές τιμές των μεταβλητών μετά την εκτέλεση των παρακάτω προγραμμάτων;

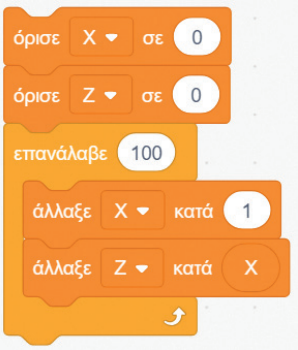
Α



Β



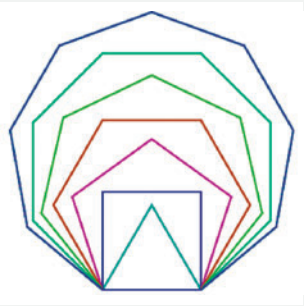
Γ

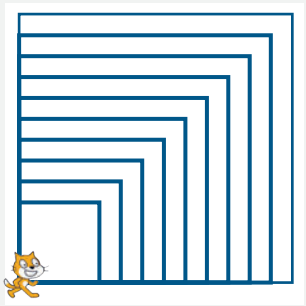


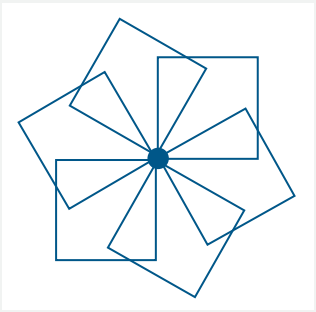
Δραστηριότητα 5

Να αναπτύξετε προγράμματα σε Scratch, τα οποία θα εμφανίζουν κάθε ένα από τα παρακάτω σχήματα.

Α Β Γ



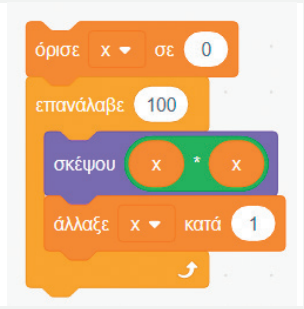




Δραστηριότητα 6

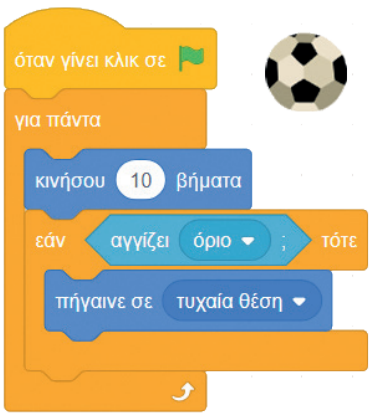
Τι υπολογίζουν και εμφανίζουν τα παρακάτω προγράμματα; Να εξηγήσετε τη λειτουργία τους.

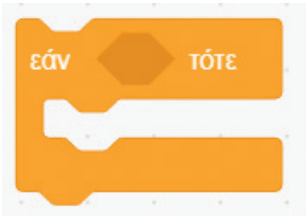
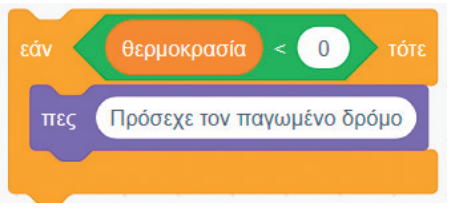
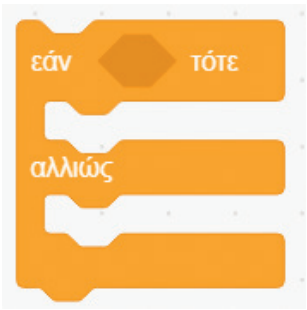
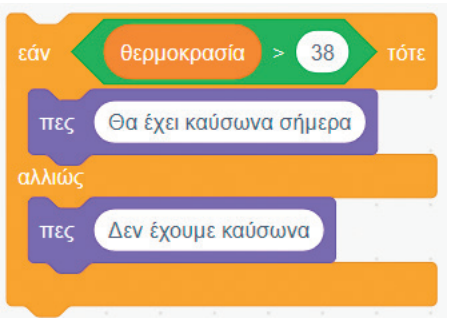
Α Β Γ



9.7 Η ώρα των αποφάσεων

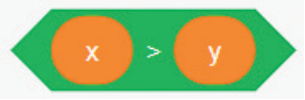
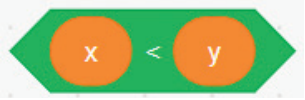
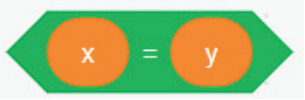
Πολλές φορές θέλουμε να ελέγξουμε διάφορες περιπτώσεις και να εκτελέσουμε συγκεκριμένες ομάδες εντολών ανάλογα με την τιμή κάποιας συνθήκης. Για αυτό χρησιμοποιούμε τις εντολές διακλάδωσης/επιλογής. Στο διπλανό παράδειγμα, μια μπάλα ποδοσφαίρου κινείται συνεχώς στο σκηνικό. Όταν αγγίζει το όριο της οθόνης, τοποθετείται αμέσως σε μια τυχαία θέση στο σκηνικό και συνεχίζει να κινείται μέχρι να φτάσει στα όρια της οθόνης κ.ο.κ. Η εντολή <πήγαινε σε τυχαία θέση> εκτελείται μόνο όταν η μπάλα αγγίξει κάποιο από τα όρια της οθόνης. Δηλαδή, κάθε φορά ελέγχεται αν ισχύει η έκφραση <αγγίζει <όριο>>.



Περιγραφή	Εντολή	Παράδειγμα
Όταν έχουμε μόνο μια περίπτωση. Οι εντολές μέσα στο τότε θα εκτελεστούν μόνο αν η συνθήκη ισχύει.		
Όταν έχουμε δύο περιπτώσεις. Αν η συνθήκη ισχύει, εκτελούνται οι εντολές μέσα στο τότε. Αν δεν ισχύει, εκτελούνται οι εντολές που αντιστοιχούν στο αλλιώς.		

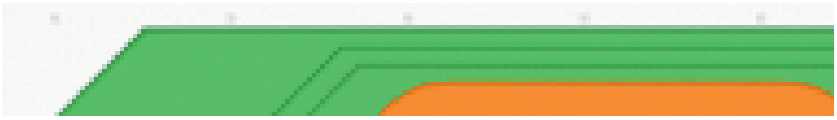
Όταν έχουμε περισσότερες από μία περιπτώσεις, χρειάζεται να χρησιμοποιήσουμε εμφωλευμένες ή ένθετες (nested) εντολές επιλογής, όπως φαίνεται στο διπλανό παράδειγμα, όπου παίρνουμε περιπτώσεις για τη θερμοκρασία. Αν η θερμοκρασία είναι θετικός αριθμός, εμφανίζεται αντίστοιχο μήνυμα. Αν όμως δεν είναι, δηλαδή η συνθήκη θερμοκρασία > 0 δεν ισχύει, τότε υπάρχουν δύο περιπτώσεις, είτε να είναι μηδέν, είτε να είναι αρνητική. Γι' αυτό το λόγο, πρέπει πάλι να χρησιμοποιήσουμε και δεύτερο έλεγχο για να διακρίνουμε ανάμεσα στις δύο αυτές περιπτώσεις. Αν δεν ισχύει ούτε η συνθήκη θερμοκρασία < 0, τότε η μόνη περίπτωση που μένει είναι η μηδενική θερμοκρασία που αντιστοιχεί στο τελευταίο αλλιώς. Άρα, στο αλλιώς φτάνουμε μόνο όταν δεν ισχύει καμία από τις συνθήκες που έχουν προηγηθεί.

Μια λογική συνθήκη μπορεί να ισχύει ή να μην ισχύει. Όταν ισχύει λέμε ότι έχει τιμή **Αληθής** ενώ όταν δεν ισχύει έχει την τιμή **Ψευδής**. Δίπλα φαίνονται οι τιμές κάποιων συνθηκών, στις οποίες εμπλέκονται οι μεταβλητές x και y με περιεχόμενο:

Συνθήκη	Αποτέλεσμα
	28 > 6, **Αληθής**
	28 < 6, **Ψευδής**
	28 = 6, **Ψευδής**

Μπορούμε να χρησιμοποιήσουμε και σύνθετες συνθήκες, αν χρειάζεται. Για παράδειγμα, αν θέλουμε η θερμοκρασία να είναι εντός συγκεκριμένων ορίων χρειάζεται η σύζευξη δύο συνθηκών με χρήση του λογικού τελεστή **και** όπως φαίνεται παρακάτω:

Μπορούμε, επίσης, να σχηματίσουμε την αντίθετη συνθήκη, δηλαδή να είναι η θερμοκρασία εκτός των παραπάνω ορίων, άρα είτε μικρότερη ή ίση από 20°C είτε μεγαλύτερη ή ίση από 30°C. Για αυτό μπορούμε να χρησιμοποιήσουμε τον λογικό τελεστή της διάζευξης (**ή**) ή τον λογικό τελεστή της άρνησης (**όχι**).



Είναι φανερό ότι η χρήση του τελεστή **όχι** όταν θέλουμε να ελέγξουμε την αντίθετη μιας λογικής συνθήκης, μπορεί να μας γλιτώσει από πολλή δουλειά και να μειώσει το μέγεθος του κώδικά μας.

Δραστηριότητα 7

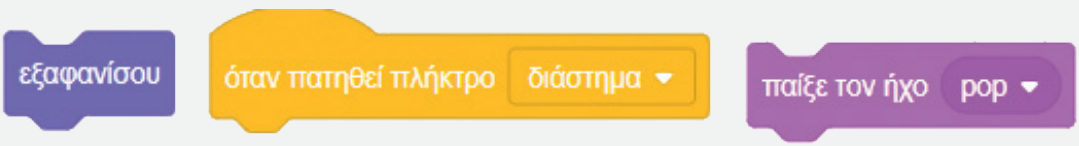
Να γράψετε ένα πρόγραμμα το οποίο θα ζητάει από τον χρήστη έναν αριθμό, θα ελέγχει αν αυτός ο αριθμός είναι άρτιος ή περιττός και θα εμφανίζει αντίστοιχο μήνυμα.

Υπόδειξη: *Να χρησιμοποιήσετε τον τελεστή υπολογισμού του υπολοίπου της ακέραιας διαίρεσης.*

Δραστηριότητα 8

Να γράψετε ένα πρόγραμμα το οποίο θα κάνει έναν συγκεκριμένο ήχο κάθε φορά που ο χρήστης θα πατάει το πλήκτρο space. Όταν ο χρήστης πατήσει το space 10 φορές το πρόγραμμα να εμφανίζει ένα μήνυμα και να εξαφανίζει τον χαρακτήρα – πρωταγωνιστή του προγράμματος.

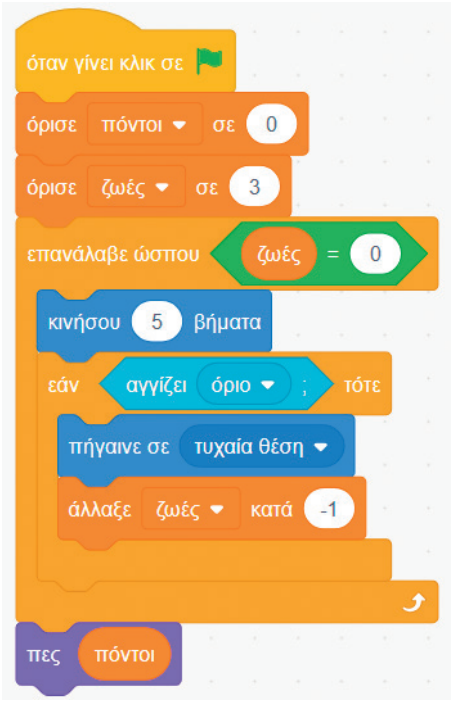
Υπόδειξη: *Μπορείτε να χρησιμοποιήσετε τις παρακάτω εντολές:*

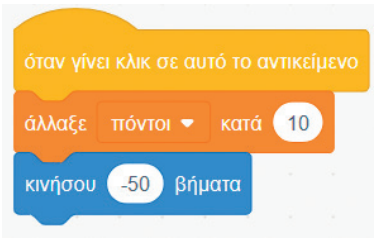


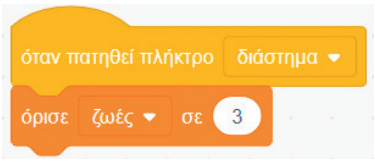
9.8 Χειρισμός γεγονότων

Μια σημαντική ομάδα εντολών, που μπορεί να αξιοποιηθεί στον σχεδιασμό παιχνιδιών, είναι τα συμβάντα (events) και οι αισθητήρες (sensing).

Θα μπορούσαμε να επεκτείνουμε το πρόγραμμα με τη μπάλα ποδοσφαίρου που κινείται συνεχώς στη σκηνή με την εξής λειτουργικότητα: Να προσπαθεί ο χρήστης-παίκτης να κάνει κλικ με το ποντίκι στην μπάλα που κινείται. Κάθε φορά που τα καταφέρνει θα παίρνει 10 πόντους, αλλά κάθε φορά που δε θα προλαβαίνει και η μπάλα θα φτάνει στο όριο της σκηνής θα χάνει μια ζωή. Για αυτό θα χρειαστούμε δύο μεταβλητές, τις ζωές και τους πόντους. Ο παίκτης ξεκινάει αρχικά με 3 ζωές και 0 πόντους. Σε περίπτωση που χάσει και την τελευταία ζωή του το πρόγραμμα τερματίζει και του εμφανίζει τους βαθμούς που συγκέντρωσε. Για αυτό χρησιμοποιούμε τη εντολή επανάληψης **Επανάλαβε ώσπου <ζωές=0>** η οποία ελέγχει σε κάθε επανάληψη αν η μεταβλητή ζωές έχει πάρει την τιμή 0.

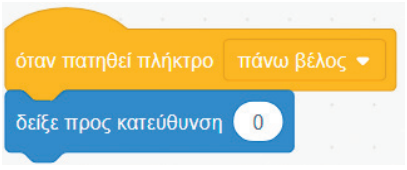


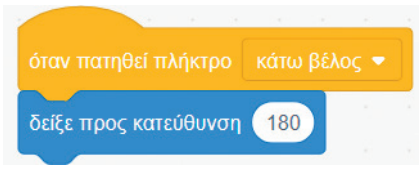


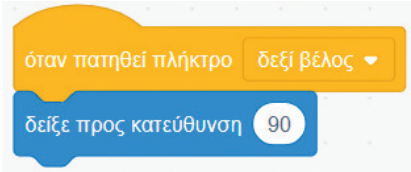


Παραπάνω προσθέσαμε και έναν ειδικό κωδικό (cheat code), ώστε όταν ο χρήστης χάνει, να μπορεί να πάρει τρεις ζωές με ένα πάτημα του πλήκτρου διάστημα (space).

Μπορούμε να γράψουμε κώδικα για τη διαχείριση και άλλων γεγονότων, όπως το πάτημα ενός πλήκτρου. Για παράδειγμα, αντί η μπάλα να κινείται τυχαία, θα μπορούσε να τη μετακινεί ο παίκτης με τα βελάκια και ένας άλλος παίκτης να χειρίζεται το ποντίκι.







Μπορείτε να δώσετε εσείς τον αντίστοιχο κώδικα για τον χειρισμό του πλήκτρου που μετακινεί τον χαρακτήρα αριστερά (αριστερό βέλος);

Προγραμματισμός οδηγούμενος από γεγονότα

Σήμερα, οι σύγχρονες εφαρμογές δεν είναι προγράμματα τα οποία ακολουθούν μια σειριακή εκτέλεση των εντολών. Αντιθέτως, αποτελούνται από πολλά μικρά προγράμματα, τα οποία είναι σχεδιασμένα να ανταποκρίνονται/χειρίζονται συγκεκριμένα γεγονότα, όπως το πάτημα ενός πλήκτρου ή του ποντικιού ή ένα μήνυμα από ένα άλλο πρόγραμμα μέσω Διαδικτύου.