1.3.5 ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ στον ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Μάθημα του προγράμματος	Μαθήματα Πληροφορικής και Τεχνολογίας
Διδακτικές ώρες	Ανάλογα με το επίπεδο των μαθητών από 4 ως 6 ώρες
Διδακτικό αντικείμενο	Εισαγωγή στον προγραμματισμό
Χρονισμός με το αναλυτικό πρόγραμμα	Β' τρίμηνο (στην Α΄ και Β΄ Λυκείου)
Διδακτική μέθοδος	Επίδειξη, προσωπική εργασία μαθητών με την καθοδήγηση του διδάσκοντος, ομαδική εργασία, ομαδική αναζήτηση λύσεων σε προβλήματα.
Διδακτικός στόχος	Οι προτεινόμενες δραστηριότητες μπορούν να χωριστούν σε 2 κατηγορίες, ανάλογα με το σκοπό που εξυπηρετούν. Οι πρώτες δραστηριότητες αποσκοπούν κυρίως στην εξοικείωση των μαθητών με το περιβάλλον προγραμματισμού και τη διαχείρισή του. Η χρήση λοιπόν βοηθητικής γλώσσας (λογικού διαγράμματος και ψευδογλώσσας) δε κρίνεται απαραίτητη. Στη δεύτερη κατηγορία εντάσσονται οι καθαυτό προγραμματιστικές δραστηριότητες. Η χρήση βοηθητικής γλώσσας είναι επιλογή του διδάσκοντος.

1. **Πήγαινε στο μενού Αρχείο και επίλεξε Άνοιγμα. Άνοιξε το αρχείο sample.x. Ανοίγουν δυο παράθυρα. Παρατήρησε τα ονόματα τους: Πηγαίος κώδικας ... και Έξοδος Προγράμματος. Στο πρώτο από αυτά θα γράφεις και θα διορθώνεις τα προγράμματα σου και στα δεύτερο θα γράφονται τα αποτελέσματα της εκτέλεσης των προγραμμάτων. Το πρόγραμμα sample.x είναι ένα τετριμμένο πρόγραμμα. Παρατήρησε ότι στο πρόγραμμα δεν υπάρχει επικεφαλίδα του είδους program xxx, ούτε και δήλωση των μεταβλητών. Υφίστανται όμως τα begin και end. Άνοιξε το μενού Μεταγλώττιση και επίλεξε Μεταγλώττιση προγράμματος. Παρατήρησε μετά από λίγο το μόνο που συμβαίνει είναι να γίνει το παράθυρο Πηγαίος κώδικας ανενεργό και το παράθυρο Έξοδος Προγράμματος ενεργό. Το σύστημα δεν έχει καμιά αντίδραση – γεγονός που σημαίνει ότι ο μεταγλωττιστής (compiler) μεταγλώττισε το πρόγραμμα χωρίς να βρει λάθη.**

Η δραστηριότητα αποσκοπεί στην πρώτη γνωριμία των μαθητών με το περιβάλλον. Μετά το άνοιγμα του αρχείου sample.x, ο καθηγητής θα πρέπει να υπογραμμίσει δυο σημεία:

α) το γεγονός ότι η γλώσσα Χ είναι ένα μικρό υποσύνολο της Pascal και επομένως τα προγράμματα των μαθητών δε θα έχουν επικεφαλίδες, ούτε και δηλώσεις μεταβλητών. Στην πορεία θα προκύψουν και άλλες διαφορές από τη Standard Pascal, τις οποίες ο καθηγητής θα πρέπει να επισημαίνει.

β) το γεγονός ότι πρέπει να γίνει μια διάκριση ανάμεσα στη μεταγλώττιση και την εκτέλεση του προγράμματος. Ο καθηγητής θα πρέπει να προειδοποιήσει τους μαθητές ότι η φαινομενική αδράνεια του συστήματος σε μια μεταγλώττιση, σημαίνει κατ' ουσία ότι η μεταγλώττιση είναι επιτυχής και το σύστημα είναι έτοιμο για εκτέλεση του προγράμματος.

2. **Εκτέλεσε το πρόγραμμα sample.x ανοίγοντας το μενού Μεταγλώττιση και επιλέγοντας την Εκτέλεση προγράμματος. Τώρα θα προκαλέσουμε μια μεταβολή στο πρόγραμμα: στην περιοχή του πηγαίου κώδικα, άλλαξε την εντολή y:=5; σε read y;. Αφού ο κώδικας του προγράμματος σου άλλαξε, πρέπει και πάλι να τον μεταγλωττίσεις. Εμφανίζεται ένα παράθυρο Αναφοράς Μεταγλώττισης, με μηνύματα. Διάβασε τα μηνύματα Προειδοποιήσεων και κλείσε το παράθυρο. Ύστερα εκτέλεσε τον επιλέγοντας Εκτέλεση προγράμματος από το μενού Μεταγλώττιση. Τι αλλάζει στη συμπεριφορά του προγράμματος;**

Η ύπαρξη δυο διακεκριμένων παραθύρων που είναι ταυτόχρονα ορατά, περιορίζει βέβαια σημαντικά το χώρο εμφάνισης των αποτελεσμάτων – μετά από ένα ορισμένο πλήθος αποτελεσμάτων ο χρήστης πρέπει να χρησιμοποιήσει τις γραμμές κύλισης για να δει τα αποτελέσματα. Ωστόσο η επιλογή αυτή διευκολύνει ιδιαίτερα τους αρχάριους. Για παράδειγμα στα περιβάλλοντα της Qbasic, της Logowriter και της Turbo Pascal (και οι τρεις γλώσσες που χρησιμοποιήθηκαν στην εκπαίδευση) δε μπορούν να συνυπάρχουν τα δυο αυτά παράθυρα δημιουργώντας σύγχυση στους μαθητές – κυρίως στην Turbo Pascal, όπου ο χρήστης μπορεί να βλέπει και τα αποτελέσματα προηγουμένων εκτελέσεων, ή τμήματα προηγουμένων εντολών σε DOS.

Ο καθηγητής θα πρέπει να επισημάνει το ρόλο της αναφοράς: εμφάνιση συντακτικών λαθών και προειδοποιήσεων – μόνο από συντακτική και όχι από λογική πλευρά. Θα πρέπει να εξηγήσει ότι ακόμη και για τα συντακτικά προβλήματα το σύστημα δε διαθέτει την ανθρώπινη νοημοσύνη και έτσι πολλές φορές τα μηνύματα του δεν ανταποκρίνονται πλήρως στην πραγματικότητα. Οι προειδοποιήσεις δε σηματοδοτούν λάθη, απλώς αναφέρουν σημεία τα οποία θα μπορούσαν να είναι λανθασμένα. Για παράδειγμα, στη συγκεκριμένη περίπτωση, η προειδοποίηση δεν έχει νόημα από τη μεριά του χρήστη, αφού ο χρήστης θα δώσει τιμή στο y την ώρα της εκτέλεσης, αλλά το σύστημα επισημαίνει το γεγονός ότι όταν μεταγλωττίζει την εντολή write y, η τιμή του y του είναι άγνωστη.

Ο καθηγητής θα πρέπει να εξηγήσει τη διαφορά που υπάρχει ανάμεσα σε ένα πρόγραμμα με απευθείας εκχώρηση τιμής και σε ένα πρόγραμμα με εισαγωγή τιμής από το πληκτρολόγιο. Θα πρέπει να κάνει μνεία της διαφορετικής οικονομίας του κάθε τρόπου λειτουργίας: ο ένας είναι πιο άκαμπτος, αλλά πολύ οικονομικός (π.χ. όταν θέλουμε να κάνουμε δοκιμές με μεγάλο αριθμό δεδομένων), ενώ ο άλλος είναι πιο ελαστικός. Ωστόσο δε θα πρέπει να επιμείνει ιδιαίτερα καθώς η προβληματική αυτή είναι ακόμη άγνωστη στους μαθητές. Η διαφορά ανάμεσα σε μια διαδραστική εκτέλεση του προγράμματος και σε μια μη-διαδραστική μπορεί να είναι πολύ σημαντική για τους μαθητές. Στα τυπικά προγραμματιστικά περιβάλλοντα οι εντολές εισόδου από το πληκτρολόγιο προκαλούν μια φαινομενική αδράνεια του συστήματος με το δρομέα ακίνητο (στην Turbo Pascal, C κλπ), ή με την εμφάνιση ενός ερωτηματικού ? (Basic) δημιουργώντας συνήθως την εντύπωση στους μαθητές ότι το σύστημα «δε λειτουργεί» πλέον ή ακόμη και ότι «χάλασε» - αποτελούν κατά κανόνα ένα από τα σημεία στα οποία οι μαθητές δυσκολεύονται να κατανοήσουν πλήρως τη λειτουργία του συστήματος. Στο περιβάλλον αυτό όμως η ρητή αίτηση από το σύστημα για απόδοση τιμής διευκολύνει την κατανόηση του διαφορετικού τρόπου λειτουργίας.

3. Εκτέλεσε το πρόγραμμα sample.x ανοίγοντας το μενού Μεταγλώττιση και επιλέγοντας την Εκτέλεση προγράμματος. Δώσε μια τιμή στη μεταβλητή ψ και παρατήρησε τη συμπεριφορά του συστήματος. Εκτέλεσε μερικές φορές ακόμη το πρόγραμμα δίνοντας κάθε φορά άλλη τιμή στην ψ. Δοκίμασε τις τιμές:

234

23.566 - πρόσεξε το δεκαδικό σημείο είναι η τελεία και όχι το κόμμα

-12.3243

12.123456789

123456789

123456789123456789

Τι παρατηρείς; Μπορείς να συνοψίσεις τα συμπεράσματα σου;

Ο καθηγητής πρέπει να επισημάνει δυο σημεία.

α) η διαφορά ανάμεσα στη μεταγλώττιση και την εκτέλεση του προγράμματος σημαίνει ότι μετά τη μεταγλώττιση ο χρήστης μπορεί να εκτελέσει όσες φορές επιθυμεί το πρόγραμμα του, εισάγοντας κάθε φορά και άλλες τιμές.

β) οι μαθητές θα είναι εξοικειωμένοι με την ιδέα ότι ο ΗΥ δεν έχει απόλυτη και απεριόριστη ακρίβεια – το γνωρίζουν ήδη από τις αριθμομηχανές τσέπης. Ο καθηγητής, χωρίς να υπεισέλθει σε λεπτομέρειες, μπορεί να επισημάνει το γεγονός ότι ο ΗΥ έχει μια περιορισμένη ακρίβεια και δυνατότητα αναπαράστασης αριθμών.

Παρόλα αυτά, το εύρος των αριθμών που δίνουν αποτελέσματα ακριβείας είναι αρκετό για τα προβλήματα που αντιμετωπίζουν συνήθως οι μαθητές στο σχολικό περιβάλλον. Ωστόσο οι μαθητές θα πρέπει να είναι ιδιαίτερα προσεκτικοί όταν κάνουν πράξεις με αριθμούς που έχουν πολλά ψηφία.

Ο καθηγητής μπορεί να δείξει στους μαθητές του με απλό τρόπο τις παρεκκλίσεις από τα ορθά αποτελέσματα που μπορεί να παράγει η περιορισμένη ακρίβεια του ΗΥ. Έτσι, για παράδειγμα, σε ένα σύστημα που διαχειρίζεται μόνο μονοψήφιους αριθμούς, οι παρακάτω πράξεις μπορούν να έχουν τα αποτελέσματα που ακολουθούν:

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	X
2	2	3	4	5	6	7	8	9	X	X
3	3	4	5	6	7	8	9	X	X	X
4	4	5	6	7	8	9	X	X	X	X
5	5	6	7	8	9	X	X	X	X	X
6	6	7	8	9	X	X	X	X	X	X
7	7	8	9	X	X	X	X	X	X	X
8	8	9	X	X	X	X	X	X	X	X
9	9	X	X	X	X	X	X	X	X	X

Οι θέσεις X του πίνακα δείχνουν τα αποτελέσματα που είναι λανθασμένα. Ανάλογα με το σύστημα κάθε X μπορεί να αντιπροσωπεύει

- ένα μήνυμα λάθους υπερχείλισης (overflow)
- ένα 9. Καθώς ο 9 είναι ο μεγαλύτερος αριθμός που μπορεί να αναπαρασταθεί από το σύστημα, το 9 είναι η καλύτερη δυνατή προσέγγιση του πραγματικού αποτελέσματος.
- Το υπόλοιπο της διαίρεσης του πραγματικού αποτελέσματος δια του 9, δηλαδή το X mod 9. Έτσι, για παράδειγμα, 8+5=4 αφού 4=13 mod 9.

4. Κάνε μερικές δοκιμές για να διαπιστώσεις τη λειτουργία του συστήματος της Αναφοράς μεταγλώττισης. Άλλαξε ένα σημείο του προγράμματος σου, μεταγλώττισε το, δες τα αντίστοιχα μηνύματα και συζήτησέ τα με τον καθηγητή σου. Δοκίμασε τις εξής αλλαγές:

Άλλαξε την εντολή y:=5; σε y=5;

Αφαίρεσε ένα ερωτηματικό ";"

Γράψε κάπου μια μεταβλητή z χωρίς εκχώρηση

Η δραστηριότητα αυτή αποσκοπεί στην εξοικείωση των μαθητών με το γεγονός ότι η ερμηνεία των μηνυμάτων και ο εντοπισμός του λάθους στον κώδικα τους δεν είναι πάντοτε προφανής και μπορεί να μην αντιστοιχεί στην αντίληψη του χρήστη για τα πιθανά λάθη που μπορεί να κάνει.

5. Από το μενού Κλάση, επίλεξε την κλάση αρχαρίων. Στο κάτω μέρος του παραθύρου του πηγαίου κώδικα εμφανίζονται μερικά εικονίδια. Διερεύνησε τη λειτουργία τους.

Τροποποίησε το πρόγραμμα σου. Σβήσε μερικές εντολές έτσι ώστε να πάρει την εξής μορφή:

x:=8;

y:=14;

z:=x+y;

write z;

Από το μενού Κλάση επίλεξε την κλάση προχωρημένων. Παρατήρησε τα νέα παράθυρα και συζήτησε με τον καθηγητή το ρόλο τους. Τρέξε το πρόγραμμα. Τρέξε το πρόγραμμα βηματικά. Κλείσε το παράθυρο της assembly και το παράθυρο των registers.

Ο καθηγητής χωρίς να προχωρήσει σε λεπτομέρειες, μπορεί να εξηγήσει το ρόλο κάθε παραθύρου: της assembly, των μεταβλητών συστήματος (registers) και των ενδιάμεσων τιμών. Στο απλό πρόγραμμα που έχουν οι μαθητές είναι εξάλλου εύκολο να δείξει την αντιστοιχία ανάμεσα στις εντολές της γλώσσας X και της γλώσσας assembly-X. Η βηματική εκτέλεση είναι ωστόσο πιο ενδιαφέρουσα καθώς επιτρέπει την παρατήρηση των διαδοχικών τιμών των μεταβλητών – έτσι το περιβάλλον γίνεται ισοδύναμο με την προσομοίωση της εκτέλεσης του προγράμματος στο χαρτί.

6. Τροποποίησε το πρόγραμμα σου και γράψε τα εξής:

x:=8;

y:=14;

z:=x+y;

x:=z+x;

x:=z+x;

y:=x+y;

write x;

write y;

write z;

Μπορείς να προβλέψεις τα αποτελέσματα που θα εμφανιστούν στο παράθυρο των αποτελεσμάτων;

Άλλαξε τις δυο πρώτες εντολές σε

read x;

read y;

Τώρα είναι δυνατόν να προβλέψεις τα αποτελέσματα που θα εμφανιστούν στο παράθυρο αποτελεσμάτων; Υπάρχει τρόπος να προβλέψεις τα αποτελέσματα αν γνωρίζεις τις τιμές που θα αποδοθούν στο x και στο y, χωρίς να κάνεις όλους τους ενδιάμεσους υπολογισμούς; Εκτέλεσε το πρόγραμμα βήμα-βήμα.

Ο υπολογισμός των τελικών αποτελεσμάτων είναι βέβαια τετριμμένος. Στην περίπτωση που οι απευθείας εκχωρήσεις αντικατασταθούν από εντολές εισόδου δεδομένων από το πληκτρολόγιο, τα τελικά αποτελέσματα δεν μπορούν βέβαια να είναι εκ των προτέρων γνωστά. Αν όμως οι αρχικές τιμές των x, y είναι αντίστοιχα a και b, τότε το αποτέλεσμα του προγράμματος μπορεί να υπολογιστεί ως μια συνάρτηση των a,b:

Περιεχόμενα μεταβλητών πριν την εκχώρηση	**Εντολή εκχώρησης**	**Περιεχόμενα μεταβλητών μετά την εκχώρηση**
x=a y=b z=?	z:=x+y	x=a y=b z=a+b
x=a y=b z=a+b	x:=x+y	x=a+a+b=2a+b y=b z=a+b
x=2a+b y=b z=a+b	x:=x+y	x=3a+2b y=b z=a+b
x=3a+2b y=b z=a+b	y:=x+y	x=3a+2b y=3a+3b z=a+b

Ο καθηγητής είναι δυνατόν να δείξει στους μαθητές τη συλλογιστική αυτή καθώς οι μαθητές δε θα βρουν δυσκολίες στην κατανόηση των στοιχειωδών αλγεβρικών πράξεων.

Το ουσιαστικό κέρδος από μια τέτοια προσέγγιση έγκειται στην ιδέα του ότι είναι δυνατόν με απλή άλγεβρα να έχουμε, σε ορισμένες περιπτώσεις, έναν έλεγχο των αποτελεσμάτων και ταυτόχρονα μια απόδειξη της εγκυρότητας τους.

Η βηματική εκτέλεση θα επιβεβαιώσει και εμπειρικά αυτό που υπολογίστηκε με τη βοήθεια της στοιχειώδους άλγεβρας. Η αξία λοιπόν της βηματικής εκτέλεσης είναι εκείνη της οικονομίας: η βηματική εκτέλεση επιτρέπει, με έναν οικονομικό τρόπο, την παρακολούθηση της τροποποίησης που υφίστανται οι τιμές των μεταβλητών και του αποτελέσματος εκτέλεσης κάθε εντολής.

7. Δυο μεταβλητές Α και Β περιέχουν δυο αριθμητικές τιμές. Επιθυμούμε να τις αντιμεταθέσουμε, δηλαδή το Α να αποκτήσει την αριθμητική τιμή του Β και το Β την αριθμητική τιμή του Α. Να γραφεί ένα πρόγραμμα που να εκτελεί την αντιμετάθεση αυτή.

Είναι η πρώτη άσκηση στην οποία η χρήση του χάρτινου ΗΥ μπορεί να έχει χρησιμότητα, καθόσον επιτρέπει την εικονική αναπαράσταση των μεταβλητών, των περιεχομένων και των μετασχηματισμών τους. Μια δυσκολία των αρχαρίων μαθητών στην επίλυση των προβλημάτων προγραμματισμού έγκειται στην κατανόηση του γεγονότος ότι οι διαθέσιμες δυνατότητες για την έκφραση της λύσης είναι μόνον αυτές που προσφέρει η χρησιμοποιούμενη γλώσσα προγραμματισμού. Επιπλέον, η σημαντικότερη ίσως δυσκολία των μαθητών έγκειται στην κατανόηση του *νέου διδακτικού συμβολαίου*, δηλαδή του πραγματικού νοήματος των προβλημάτων που τίθενται και της ζητούμενης λύσης. Συγκεκριμένα, οι μαθητές πολλές φορές εκλαμβάνουν το παράδειγμα ως το ζητούμενο. Αν ο καθηγητής δώσει ως παράδειγμα αρχική τιμή στο Α το 8 και στο Β το 5, οι μαθητές προτείνουν ως λύση του προβλήματος τις εκχωρήσεις Α:=5 και Β:=8.

Μια εκδοχή λύσης είναι λοιπόν η εξής:

```
A:=8;
B:=5;  (αυτά προτείνει ο καθηγητής ως αρχική κατάσταση)
A:=5;
B:=8:  (η λύση που προτείνουν οι μαθητές)
```

Ο καθηγητής ανατρέχοντας στην εκφώνηση, υπενθυμίζει ότι η λύση πρέπει να είναι σωστή για δυο τυχαίους αριθμούς. Με άλλα λόγια, πρέπει την αντιμετάθεση να την κάνει ο ΗΥ και άρα ο πραγματικός στόχος του προγραμματιστή, δηλαδή των μαθητών, είναι η διατύπωση εκείνων των εντολών που θα επιτρέψουν στον ΗΥ να αντιμεταθέσει δυο οιεσδήποτε τιμές – μέσα στα όρια των αποδεκτών αριθμών. Επομένως αν η αρχική κατάσταση είναι Α=100 και Β=200, τότε το πρόγραμμα θα είναι:

```
A:=100;
B:=200;
A:=5;
B:=8:
```

Οι μαθητές μπορούν στη συνέχεια να κατανοήσουν και το γεγονός ότι οι εκχωρήσεις Α:=Β και Β:=Α (που πολλές φορές προτείνονται ως λύση) δε δίνουν το επιθυμητό αποτέλεσμα.

Είναι πολύ πιθανό ορισμένοι μαθητές να προτείνουν τη σωστή λύση προτείνοντας μια τρίτη ενδιάμεση μεταβλητή – όπως ακριβώς για να αντιμεταθέσουμε τα (υγρά) περιεχόμενα δυο δοχείων χρειαζόμαστε και ένα τρίτο, βοηθητικό δοχείο.

Άρα μια λύση είναι:

```
Temp:= A;
A:=B;
B:=Temp;
```

Ο καθηγητής μπορεί να ρωτήσει, αν είναι δυνατόν να επινοηθεί μια λύση χωρίς χρήση της Temp. Πιθανότατα οι μαθητές θα απαντήσουν αρνητικά, οπότε ο καθηγητής μπορεί να προτείνει το εξής:

```
A:=A+B;
B:=A-B;
A:=A-B;
```

Μια εκτέλεση του προγράμματος δείχνει σωστά αποτελέσματα. Η παρακολούθηση του παράθυρου με τις ενδιάμεσες τιμές των μεταβλητών δείχνει την εξέλιξη των διαδοχικών τιμών.

Ο καθηγητής μπορεί και πάλι να προτείνει μια αλγεβρική ερμηνεία του αποτελέσματος:

Περιεχόμενα μεταβλητών πριν την εκχώρηση		**Εντολή εκχώρησης**	**Περιεχόμενα μεταβλητών μετά την εκχώρηση**	
A=x	B=y	A:=A+B	A=x+y	B=y
A=x+y	B=y	B:=A-B	A:=x+y	B=x+y-y=x
A:=x+y	B=x	A:=A-B	A=x+y-x=y	B=x

Ο καθηγητής μπορεί ακόμη να ρωτήσει αν υπάρχει κάποιο μειονέκτημα στη δεύτερη αυτή πιο οικονομική μέθοδο. Μπορεί ακόμη να ρωτήσει αν η δεύτερη μέθοδος «καλύπτει» το ίδιο σύνολο αριθμών με την πρώτη. Προφανώς, αν οι αριθμοί που εκχωρούνται στα Α, Β έχουν μεγάλο αριθμό ψηφίων η δεύτερη μέθοδος, επειδή προσθέτει τις τιμές αυτές, έχει μεγάλες πιθανότητες να παρουσιάσει πρόβλημα υπερχείλισης.

Ακόμη ο καθηγητής μπορεί να συζητήσει το θέμα της αποτελεσματικότητας (ταχύτητας) της μιας και της άλλης μεθόδου. Αν υποτεθεί ότι όλες οι πράξεις απαιτούν τον ίδιο χρόνο Τ, για να εκτελεστούν, τότε η πρώτη μέθοδος απαιτεί πιο μικρό χρόνο αφού (όπως διαπιστώνεται από το πλήθος εντολών της assembly που αντιστοιχούν στις εντολές εκχώρησης και στις δυο περιπτώσεις), η πρώτη μέθοδος απαιτεί 6 στοιχειώδεις πράξεις, ενώ η δεύτερη δώδεκα.

Η δεύτερη αυτή τεχνική για την αντιμετάθεση των δυο τιμών, θα μπορούσε να πραγματοποιηθεί με ταχύτερο τρόπο, αν, παρεμβαίνοντας στον κώδικα assembly, παραλειφθούν ορισμένες ενδιάμεσες εκχωρήσεις οι οποίες είναι προφανώς περιττές.

8. Η «αριθμητική» της γλώσσας Χ είναι αρκετά περιοριστική: επιτρέπει μόνο πράξεις μεταξύ ακεραίων. Το μεγαλύτερο εμπόδιο φαίνεται να παρουσιάζεται στη διαίρεση: πως μπορούμε να κάνουμε διαίρεση που δεν είναι τέλεια και να πάρουμε σωστό αποτέλεσμα; Πιο συγκεκριμένα, επιθυμούμε δυο ειδών διαιρέσεις:

Διαίρεση ευκλείδεια: για παράδειγμα 7 δια 2 να πάρουμε πηλίκο 3 και υπόλοιπο 1.

Διαίρεση με δεκαδικά: για παράδειγμα 7 δια 2 να πάρουμε πηλίκο 3,5.

Μπορούμε να κάνουμε τέτοιου είδους διαιρέσεις;

Η απάντηση είναι βέβαια θετική και στις δυο περιπτώσεις:

```
begin
read x;
read y;
p:=x/y;
r:=x-p*y;
write p;
write r;
end.
```

```
begin
read x;
read y;
p:=x/y;
r:=x-p*y;
decimal_digits:=r*1000/y;
write p;
write decimal_digits;
end.
```

9. Δίνεται ένα ποσόν σε δραχμές και θέλουμε να το εκφράσουμε χρησιμοποιώντας τον ελάχιστο αριθμό νομισμάτων. Για παράδειγμα, το ποσόν των 17845 δραχμών απαιτεί 1 χαρτονόμισμα των 10000, 1 των 5000, 2 των 1000, ένα των 500, 3 των 100 δραχμών κοκ. Μπορούμε να πετύχουμε την ανάλυση αυτή με τη βοήθεια του ΗΥ, εισάγοντας απλώς το ποσόν σε δραχμές;

Ουσιαστικά χρησιμοποιούμε τα υπόλοιπα των ευκλείδειων διαιρέσεων που χρησιμοποιήσαμε και στην προηγούμενη άσκηση. Ο κώδικας είναι αρκετά μακρύς – θα ήταν πολύ σύντομος αν χρησιμοποιούντο πίνακες – αλλά το αποτέλεσμα είναι εντυπωσιακό:

```
begin
read x;
dekax:=x/10000;
rest:=x-dekax*10000;
pentox:=rest/5000;
rest:=rest-pentox*5000;
xil:=rest/1000;
rest:=rest-xil*1000;
pentak:=rest/500;
rest:=rest-pentak*500;
ekatost:=rest/100;
rest:=rest-ekatost*100;
penint:=rest/50;
rest:=rest-penint*50;
eikos:=rest/20;
rest:=rest-eikos*20;
dekariko:=rest/10;
rest:=rest-dekariko*10;
taliro:=rest/5;
rest:=rest-taliro*5;
drachmes:=rest;
write dekax;
write pentox;
```

```
write xil;
write pentak;
write ekatost;
write penint;
write eikos;
write dekariko;
write taliro;
write drachmes;
end.
```

10. Να γραφεί ένα πρόγραμμα το οποίο από δυο δεδομένους αριθμούς εμφανίζει στην Έξοδο του προγράμματος τον μικρότερο από τους δυο – ή ένα από τους δυο αν οι αριθμοί είναι ίσοι. Στη συνέχεια δίνονται τρεις αριθμοί. Επιθυμούμε να εμφανιστεί στην έξοδο του προγράμματος ο μικρότερος από αυτούς.

Θα μπορούσαν να υιοθετηθούν δυο διαφορετικές προσεγγίσεις:

```
if y>x then write x;
if x>y then write y;
if x=y then write x;
```

ή ακόμη

```
min:=x
if x>y then min:=y
write min;
```

Η δραστηριότητα αυτή αποσκοπεί στη κατανόηση της λειτουργίας της εντολής επιλογής if...then. Το πρώτο τμήμα της είναι τετριμμένο, το δεύτερο όμως απαιτεί μια συστηματική αντιμετώπιση όλων των περιπτώσεων.

```
min:=x;
if min>y then min:=y;
if min>z then min:=z;
write min;
```

Η χρήση ενός λογικού διαγράμματος βοηθά στην περίπτωση αυτή στην κατανόηση και την οργάνωση της αντιμετώπισης όλων των περιπτώσεων. Η επαλήθευση με διάφορες τριάδες αριθμών θα επιβεβαιώσει και εμπειρικά την ορθότητα του προγράμματος.

Το ενδιαφέρον ερώτημα είναι βέβαια πως θα εντοπίσουμε – μάλλον πως το πληροφορικό σύστημα θα εντοπίσει – τον ελάχιστο από ένα «μεγάλο» σύνολο αριθμών, άνω των 3. Ο καθηγητής μπορεί να παραπέμψει σε κατοπινή δραστηριότητα.

11. Δίνεται από το πληκτρολόγιο μια ηλικία (από 1 έως 95 ετών). Επιθυμούμε ο ΗΥ να εμφανίσει ένα 0 στο παράθυρο εξόδου, αν η ηλικία είναι μικρότερη από 30 έτη, να εμφανίσει ένα 1 αν η ηλικία είναι 30 έτη ή άνω των 30 αλλά μικρότερη των 60 ετών και να εμφανίσει ένα 2 στις υπόλοιπες περιπτώσεις.

Η λύση και πάλι είναι τετριμμένη. Ωστόσο πολλοί μαθητές έχουν την τάση να θεωρούν ότι στις εντολές που δίνουν προς τον ΗΥ – όπως ακριβώς και σε μια ανθρώπινη επικοινωνία – υπάρχουν περιπτώσεις που δεν αναφέρονται ρητά γιατί υπονοούνται. Έτσι, αντί για ένα ορθό πρόγραμμα όπως το παρακάτω:

```
If 30>age write 0;
If 60>age then if age>29 then write 1;
If age>59 then write 2;
```

οι μαθητές μπορεί να παράγουν προγράμματα όπως αυτό που ακολουθεί:

```
if 30>age then write 0;
if 60>age then write 1;
write 2;
```

Αν η απάντηση στην πρώτη «ερώτηση» είναι αρνητική, τότε (σε μια ανθρώπινη συνομιλία) η δεύτερη «ερώτηση» αντιστοιχεί σε ένα εύρος ηλικιών ανάμεσα στα 30 και τα 60: *είναι κάτω των 30; Μήπως είναι κάτω των 60;* Παρόμοια και η τελευταία εντολή πρέπει να ερμηνευθεί μάλλον μέσα στα πλαίσια μιας ανθρώπινης επικοινωνίας, παρά μέσα στα πλαίσια μαις επικοινωνίας ανθρώπου και ΗΥ.

12. Να βρεθεί το άθροισμα και το γινόμενο τριών αριθμών.

Ο καθηγητής θα πρέπει να τονίσει και πάλι τη γενικότητα του προβλήματος: το αληθινό πρόβλημα δεν είναι ο προσδιορισμός του αθροίσματος τριών συγκεκριμένων αριθμών, αλλά τριών αριθμών οι οποίοι θα προσδιοριστούν τη στιγμή της εκτέλεσης του προγράμματος. Αλλά στο πρόβλημα αυτό τίθεται και ένα γενικότερο ερώτημα. Είναι αρκετά σπάνιο να αντιμετωπίσουμε το πρόβλημα του υπολογισμού του αθροίσματος 3 και μόνον αριθμών. Αντίθετα υπάρχει ενδεχόμενο να αντιμετωπίσουμε το πρόβλημα της άθροισης πολλών αριθμών (για παράδειγμα των βαθμών ενός τριμήνου προκειμένου να υπολογίσουμε το μέσο όρο της βαθμολογίας ενός μαθητή). Το ουσιαστικό λοιπόν ερώτημα είναι το εξής: υπάρχει τρόπος ώστε το σύστημα να μας βοηθήσει να υπολογίσουμε το άθροισμα πολλών αριθμών; Είναι ένα ερώτημα το οποίο μπορεί να απαντηθεί θετικά με τη χρήση των επαναληπτικών διαδικασιών.

Εισαγωγή στις επαναληπτικές διαδικασίες και δομές

Επειδή οι επαναληπτικές δομές είναι πολύ σημαντικές, υπενθυμίζονται συνοπτικά τα κυριότερα σημεία που τις αφορούν.

Μια επαναληπτική διαδικασία έχει τη γενική μορφή:

```
<Αρχικοποίηση>
<ΕΠΑΝΑΛΗΨΗ>
        <ομάδα εντολών 1>
        if <συνθήκη εξόδου> then goto xxx
        <ομάδα εντολών 2>
        if <συνθήκη εξόδου 2> then goto yyy
        .....
```

<ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ>

Οι συμβολικές εκφράσεις <ΕΠΑΝΑΛΗΨΗ> και <ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ> σημειώνουν απλώς την αρχή και το τέλος των εντολών που πρέπει να εκτελεστούν. Η συνθήκη εξόδου έχει το εξής νόημα: αν η συνθήκη είναι αληθής, τότε η ροή του προγράμματος μεταφέρεται στο σημείο με επιγραφή xxx. Αλλιώς εκτελείται η <ομάδα εντολών 2>.

Το γενικευμένο αυτό σχήμα μπορεί να εξειδικευθεί ως εξής:

- Αν <ομάδα εντολών 1> είναι κενή και xxx είναι ακριβώς μετά το <ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ>, τότε η δομή αυτή παριστάνει μια δομή while condition do begin…end
- Αν <ομάδα εντολών 2> είναι κενή και xxx είναι ακριβώς μετά το <ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ>, τότε η δομή αυτή παριστάνει μια δομή repeat…until condition
- Αν στην αρχικοποίηση υπάρχει μια εντολή M<= value και στην <ομάδα εντολών 1> ή στην <ομάδα εντολών 2> μια εντολή αύξησης M<=M+increment και επιπλέον η συνθήκη ελέγχει την τιμή του M, τότε η δομή αντιστοιχεί με μια δομή

  for M:=value to…do begin….end

Με ανάλογες θεωρήσεις, μπορούμε να αντιστοιχίσουμε τη δομή αυτή με όλες τις επαναληπτικές δομές που χρησιμοποιούνται στις γλώσσες προγραμματισμού.

Ποιο ακριβώς είναι το *πεδίο εγκυρότητας* κάθε επαναληπτικής δομής – δηλαδή σε ποιες περιπτώσεις χρησιμοποιείται κάθε μία;

Ένα μεγάλο μέρος των προβλημάτων που αντιμετωπίζονται προγραμματιστικά, ανήκουν στην κατηγορία των προβλημάτων *σειριακής επεξεργασίας*.

Ονομάζουμε σειρά μια διαδοχή στοιχείων $<\Sigma_1\ \Sigma_2\ \Sigma_3\ \ldots\ \Sigma_v > T$ - η διαδοχή είναι ουσιαστικά μια απεικόνιση του συνόλου των Σ_i στους φυσικούς. Η σειριακή επεξεργασία έγκειται στην εφαρμογή ενός τυποποιημένου σχήματος:

Αρχικές επεξεργασίες

Προσπέλαση πρώτου στοιχείου

Εφόσον το τρέχον στοιχείο δεν είναι το τελευταίο

{Επεξεργασία τρέχοντος στοιχείου

Προσπέλαση επόμενου στοιχείου}

Τελικές επεξεργασίες

Το γενικευμένο αυτό σχήμα έχει ευρύτατες εφαρμογές. Για παράδειγμα η «προσπέλαση» στοιχείου μπορεί να αντιστοιχεί σε μια μεγάλη ποικιλία δυνατοτήτων: το επόμενο στοιχείο μπορεί να είναι το επόμενο σε ένα σειριακό αρχείο, σε έναν πίνακα, σε μια λίστα, σε μια ακολουθία αριθμών κά. Στο σχήμα αυτό, το τελευταίο στοιχείο T δεν ανήκει στη διαδοχή, δεν αποτελεί δηλαδή μέρος της ακολουθίας. Αν όμως το T ταυτίζεται με το Σ_v, τότε μπορεί να εφαρμοστεί το παρακάτω σχήμα:

Αρχικές επεξεργασίες

{Προσπέλαση επόμενου στοιχείου

Επεξεργασία τρέχοντος στοιχείου}

΄Εως ότου προσπελαθεί το τελευταίο στοιχείο

Τελικές επεξεργασίες

Το πρώτο σχήμα αντιστοιχεί σε μια δομή while, ενώ το δεύτερο σε μια δομή repeat…until.

Στην περίπτωση που το N είναι γνωστό, μια κατάλληλη δομή είναι βέβαια η for…

Για τους μαθητές που κάνουν τα πρώτα βήματα στον προγραμματισμό, όλη αυτή η προβληματική, στη γενικότητά της, δεν έχει κανένα νόημα. Ορισμένα όμως σημεία της, είναι σημαντικά για να κατανοήσουν οι μαθητές το μηχανισμό των επαναληπτικών δομών.

- Στις διάφορες δομές, φαίνεται πως υφίσταται μια ιεραρχία «δυσκολίας» όσον αφορά την χρήσή τους. Διεθνείς έρευνες φαίνεται να δείχνουν ότι η παρακάτω κλίμακα παρουσιάζει μια σειρά δομών με αυξανόμενο «βαθμό δυσκολίας κατανόησης και ορθής χρήσης»

  Χρήση του goto

  Repeat N times (δομή που χρησιμοποιείται στη γλώσσα Logo)

  Repeat … until

  For ….begin..end

  While..begin…end

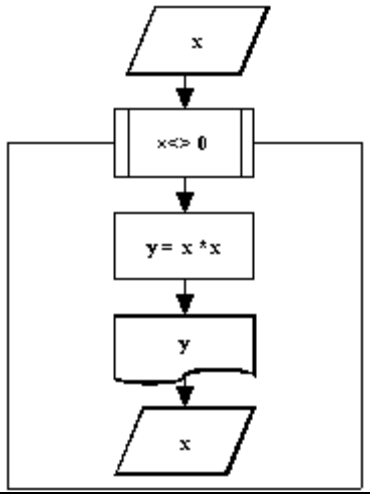
Στο περιβάλλον ΔΕΛΥΣ θα χρησιμοποιήσουμε την εντολή While..begin…end – καθώς είναι η ευρύτερα χρησιμοποιούμενη εντολή στις σύγχρονες γλώσσες προγραμματισμού (και κατέχει μια ανάλογη θέση στο σχολικό βιβλίο). Όπου χρειαστεί, θα προσομοιώσουμε τη λειτουργία των άλλων δομών (κυρίως της For ….begin..end) με τη βοήθεια της While..begin…end.

Γιατί η εντολή αυτή είναι η πιο δύσκολη; Θεωρούμε ότι υφίστανται δυο τουλάχιστον λόγοι: ο πρώτος και σημαντικότερος είναι συνδεδεμένος με την ίδια τη φύση της δομής. Η ανάγκη για πραγματοποίηση ενός τεστ, πριν ακόμη από την εισαγωγή και επεξεργασία των δεδομένων, είναι ασυνήθιστη στην καθημερινή μας πρακτική. Η “απόσταση” αυτή φαίνεται να είναι πηγή των δυσκολιών που συναντούν οι αρχάριοι στη χρήση της δομής αυτής. Όπως είναι φυσικό, η δομή while είναι συνδεδεμένη με μια προβληματική και η χρήση της έχει νόημα σε συγκεκριμένες περιπτώσεις – και εκεί βρίσκεται η δεύτερη πιθανή αιτία δυσκολιών για τους αρχάριους προγραμματιστές: η χρήση της πρέπει να είναι φανερή μέσα στα πλαίσια επίλυσης συγκεκριμένων προβλημάτων – αλλιώς μοιάζει να είναι μια αναίτια πολύπλοκη δομή.

Στις δραστηριότητες που προτείνονται στους μαθητές, εισάγονται προοδευτικά τα στοιχεία της χρήσης των δομών επανάληψης.

13. Να γραφεί ένα πρόγραμμα το οποίο θα δέχεται αριθμούς από το πληκτρολόγιο. Για κάθε αριθμό που θα εισάγει ο χρήστης, το πρόγραμμα θα εμφανίζει το τετράγωνο του. Το πρόγραμμα θα σταματάει όταν ο χρήστης εισάγει το 0.

Επειδή τα πρώτα αυτά προγράμματα είναι τελείως ασυνήθιστα για τους αρχάριους προγραμματιστές, είναι προτιμότερο ο διδάσκων να επιδείξει τη λειτουργία τους. Ενδεικτικά στα πρώτα αυτά προγράμματα παραθέτουμε τον αλγόριθμο σε μια ψευδογλώσσα, με λογικό διάγραμμα (για εφαρμογή στο χάρτινο υπολογιστή) και στη γλώσσα-Χ. Έτσι ο διδάσκων μπορεί να δείξει το πρόγραμμα:

Διάβασε το x
Όσο ο αριθμός δεν είναι 0
Υπολόγισε το τετράγωνο του
Εμφάνισε το τετράγωνο
Διάβασε το επόμενο x
		begin
		read x;
		while x <> 0 do
		begin
		y:=x*x;
		write y;
		read x;
		end;
		end.

Ο διδάσκων πρέπει πρώτα να εκτελέσει το πρόγραμμα “με το χέρι”, στο χάρτινο υπολογιστή, έτσι ώστε οι μαθητές να αποκτήσουν πλήρη έλεγχο των αναμενόμενων αποτελεσμάτων.

Η βηματική εκτέλεση του προγράμματος επιτρέπει την παρατήρηση των διαδοχικών τιμών των μεταβλητών. Ο διδάσκων πρέπει να τονίσει το γεγονός ότι

- το πρόγραμμα λειτουργεί σωστά για οποιοδήποτε πλήθος δεδομένων. Ανεξάρτητα από το πότε ο χρήστης θα αποφασίσει να διακόψει τη λειτουργία του, εισάγοντας 0, το πρόγραμμα παράγει σωστά αποτελέσματα
- το πρόγραμμα λειτουργεί σωστά ακόμη και αν ο χρήστης εισάγει κατευθείαν το 0.
- η μεταβλητή x έχει μια διπλή λειτουργία: παριστάνει και τον αριθμό που θα υψωθεί στο τετράγωνο και μια *μεταβλητή ελέγχου*.

14. Τι αλλαγές πρέπει να κάνετε στο παραπάνω πρόγραμμα έτσι ώστε να εμφανίζεται όχι το τετράγωνο, αλλά ο κύβος κάθε εισαγόμενου αριθμού; Τι αλλαγές πρέπει να κάνετε στο παραπάνω πρόγραμμα έτσι ώστε να εμφανίζεται το τετράγωνο και ο κύβος κάθε εισαγόμενου αριθμού;

Οι τροποποιήσεις αυτές στοχεύουν στην εξοικείωση των μαθητών με τη λειτουργία της επαναληπτικής δομής. Σε κάθε περίπτωση, ο διδάσκων εκτελεί με το χέρι στο χάρτινο υπολογιστή το πρόγραμμα και το κατόπιν με βηματική εκτέλεση.

15. Να γραφεί ένα πρόγραμμα, έτσι ώστε να αποδοθεί σε μια μεταβλητή x ένας ακέραιος, έστω ο 2, και στη συνέχεια ο ΗΥ να εμφανίσει 10 φορές το περιεχόμενο της μεταβλητής x. Στη συνέχεια να τροποποιηθεί το πρόγραμμα έτσι ώστε ο αριθμός να εισάγεται από το πληκτρολόγιο. Τέλος να τροποποιηθεί το πρόγραμμα έτσι ώστε ο αριθμός των εμφανίσεων της τιμής του x να είναι επίσης μεταβλητός (να εισάγεται από το πληκτρολόγιο)

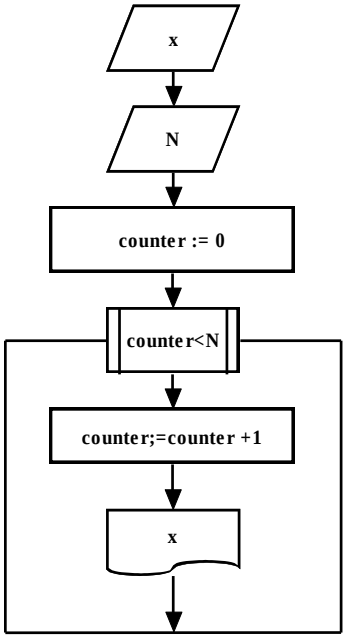
Η προφανής λύση είναι βέβαια να γραφεί 10 φορές η εντολή write x;

Ωστόσο υπάρχει και μια οικονομικότερη λύση: η χρήση ενός βρόχου. Από την εισαγωγή που προηγήθηκε, προκύπτει ότι η οικονομικότερη λύση έγκειται στη χρήση ενός μετρητή counter.

Ο διδάσκων θα πρέπει να τονίσει ιδιαίτερα τον “ασυνήθη” ρόλο της μεταβλητής counter. Είναι μια μεταβλητή “αόρατη” (δεν εμφανίζεται στα αποτελέσματα), η οποία εισάγεται αποκλειστικά και μόνο για να ελέγχει το βρόχο. Με άλλα λόγια η προγραμματιστική λύση του προβλήματος απαιτεί την εισαγωγή πρόσθετων μεταβλητών, οι οποίες δεν είναι “εμφανείς” στην εκφώνηση του προβλήματος – και αυτό είναι ένα συχνό φαινόμενο στον προγραμματισμό, όπου πολλές φορές η προγραμματιστική λύση διαφέρει ουσιωδώς από τη λύση “με το χέρι”.

Ο διδάσκων θα πρέπει επίσης να επισημάνει τη λεπτή ισορροπία μεταξύ αρχικής τιμής και της συνθήκης ελέγχου του βρόχου: ο counter έχει αρχική τιμή 0 και η συνθήκη συνεχείας του βρόχου είναι counter < 10. Πώς θα ανταποκριθεί ο ΗΥ αν μεταβληθούν οι τιμές αυτές (πρόκειται ουσιαστικά για τις άλλες δραστηριότητες που προτείνονται);

```
Διάβασε το x
Διάβασε το N
Θέσε 0 στο μετρητή
Όσο ο μετρητής δεν είναι N
        αύξησε το μετρητή
        τύπωσε το x
```

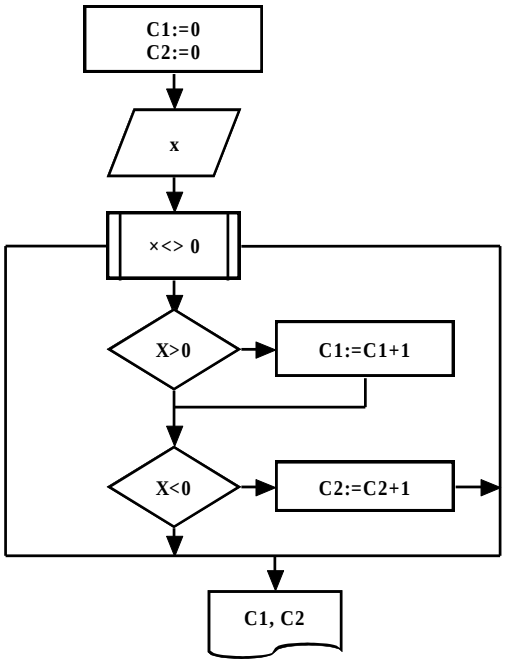



```
begin
counter:=0;
x:=2;
while 10 > counter do
  begin
  counter:=counter + 1;
  write x;
  end;
end.
```

16. Να γραφεί ένα πρόγραμμα, έτσι ώστε ο χρήστης να εισάγει από το πληκτρολόγιο μια σειρά ακεραίων. Η εισαγωγή θα σταματάει όταν ο χρήστης εισάγει το 0. Όταν σταματήσει η εισαγωγή, ο ΗΥ θα εμφανίζει το πλήθος των θετικών και των αρνητικών που εισήγαγε ο χρήστης.

Προφανώς για την επίλυση του προβλήματος απαιτούνται δυο ξεχωριστοί μετρητές (C1 και C2), οι οποίοι θα καταμετρούν το πλήθος των θετικών και των αρνητικών αριθμών. Ο διδάσκων πρέπει να τονίσει το γεγονός ότι η επιλογή του πλήθους και του "είδους" των μεταβλητών εξαρτάται αποκλειστικά από το πρόβλημα.

Θέσε C1=0 και C2=0
Διάβασε το x
Όσο το x δεν είναι 0
  Αν το x<0 αύξησε C2
  Αν το x>0 αύξησε C1
  Διάβασε x
Τύπωσε C1, C2




```
begin
C1:=0;
C2:=0;
read x;
while x<>0 do
  begin
  if x>0 then C1:=C1+1;
  if 0>x then C2:=C2+1;
  end;
write C1;
write C2;
end.
```

17. Να γραφεί ένα πρόγραμμα, έτσι ώστε ο χρήστης να εισάγει μια σειρά αριθμών, με τελευταίο το 0. Ο ΗΥ θα πρέπει να εμφανίζει το πλήθος των αριθμών που είναι ανάμεσα σε δυο δοσμένους αριθμούς Α και Β, το πλήθος αυτών που είναι μικρότεροι από τον Α και αυτών που είναι μεγαλύτεροι από το Β (υποτίθεται ότι Α<Β).

Το πρόβλημα ουσιαστικά απαιτεί τρεις αθροιστές. Προσοχή στα τεστ – έτσι ώστε οι μετρητές να λειτουργούν σωστά στις περιπτώσεις που ο εξεταζόμενος αριθμός συμπίπτει με το Α ή το Β.

Διάβασε το Α και το Β
Διάβασε το Χ
Οσο το Χ δεν είναι μηδέν
  Αν το Χ είναι μικρότερο του Α τότε αύξησε C1
  Αν το Χ είναι μεγαλύτερο του Β τότε αύξησε C3
  Αν Χ μεγαλύτερο ή ίσο του Α τότε
    αν Χ μικρότερο ή ίσο του Β τότε αύξησε C2
Τύπωσε τα C1, C2, C3

```
Begin
C1:=0; C2:=0; C3:=0;
read A; read B; read x;
while x<>0 do
  begin
  if A > x then C1:=C1+1
  if x > B then C3:=C3=1;
  if x > A-1 then if  B+1 > x then C2:=C2+1;
  read x;
  end;
write C1; write C2; write C3;
end.
```

18. Να γραφεί ένα πρόγραμμα, έτσι ώστε αν ο χρήστης εισάγει από πληκτρολόγιο ένα φυσικό αριθμό Ν, ο ΗΥ να εμφανίσει όλους τους φυσικούς από 0 ως Ν και τα τετράγωνα τους. Να τροποποιηθεί το πρόγραμμα έτσι ώστε αν ο χρήστης εισάγει από το πληκτρολόγιο δυο αριθμούς Α και Β, τότε ο ΗΥ να εμφανίσει όλους τους φυσικούς που είναι ανάμεσα τους (συμπεριλαμβανομένων των Α και Β) και τα τετράγωνα τους. Το πρόγραμμα σας λαμβάνει υπόψη το γεγονός ότι μπορεί οι Α και Β να δοθούν με αντίστροφη σειρά;

Το σημαντικότερο ίσως πρόβλημα που θα αντιμετωπίσει ο διδάσκων είναι εκείνο της εκτύπωσης:

- Αν η διαφορά των Α και Β είναι μεγάλη, τότε τα αποτελέσματα δε θα είναι όλα ορατά ταυτοχρόνως στην οθόνη.
- Η εμφάνιση των αποτελεσμάτων θα είναι “ασυνήθιστη”, καθώς θα εμφανίζονται σε μια στήλη

Και τα δυο αυτά στοιχεία μπορεί να προκαλέσουν σύγχυση στους μαθητές – ο διδάσκων θα πρέπει να εξηγήσει το αληθινό πρόβλημα – που είναι το format εκτύπωσης και όχι η ορθότητα του προγράμματος.

Θα πρέπει να δοθεί επίσης προσοχή στο γεγονός ότι πιθανόν οι Α, Β να δοθούν με αντίστροφη σειρά και θα πρέπει να ληφθεί η σχετική πρόνοια.

```
Διάβασε Α, Β
Αν Α > Β τότε
        Θέσε Τ = Α
        Θέσε Α=Β
        Θέσε Β=Τ
Θέσε CheckVal = Α
Όσο CheckVal είναι μικρότερο ή ίσο του Β
     Τύπωσε το CheckVal
      Τύπωσε το τετράγωνο του CheckVal
      Αύξησε το CheckVal μια μονάδα
```

```
begin
read A; read B;
if A > B then
  begin
  T:=A;
  A:=B;
  B:=T;
  End;
CheckVal:=A;
while B+1 > CheckVal  do
  begin
  write CheckVal;
  square:=CheckVal * CheckVal;
  write square;
  CheckVal:=CheckVal + 1;
  End;
End.
```

19. Από μια σειρά αριθμών να προσδιοριστεί ο ελάχιστος.

Η δραστηριότητα αυτή, παρά τη φαινομενική της απλότητα, ουσιαστικά εισάγει τους μαθητές σε ουσιαστικά θέματα του προγραμματισμού.

Το πρώτο αξιοσημείωτο στοιχείο του προβλήματος, έγκειται στην έκφραση "*από μια σειρά αριθμών*".

Στις συνήθεις καταστάσεις, η εύρεση ενός ακρότατου από μια σειρά αριθμών (του ελαχίστου ή του μεγίστου) δε θέτει κάποιο ιδιαίτερο πρόβλημα: αρκεί να ρίξει κανείς μια ματιά στα δεδομένα και να εντοπίσει το ζητούμενο στοιχείο. Ωστόσο, το πρόβλημα γίνεται λίγο πιο περίπλοκο, αν το πλήθος των δεδομένων είναι πολύ μεγάλο, όπως στο παρακάτω παράδειγμα (σας συμβουλεύουμε να δημιουργήσετε μια ανάλογη σελίδα ή να την φωτοτυπήσετε και να τη διανείμετε στους μαθητές σας):

53	734	979	982	61	520	399	140	437	526	184	894	706	835	968	111	692	663	312	973	903	508	439	245	153	665	959
768	41	435	300	531	605	196	225	756	68	721	89	797	654	751	495	180	551	273	782	175	69	787	384	119	662	783
62	587	667	215	77	341	312	68	348	328	225	624	556	894	107	917	886	902	105	36	195	160	795	445	977	666	341
835	304	350	657	514	52	461	212	399	269	960	160	195	114	37	312	681	374	716	173	204	22	213	611	257	552	228
536	231	199	476	496	189	365	780	242	728	721	567	236	968	465	687	39	257	44	26	803	793	407	382	248	779	727
410	377	133	617	642	925	690	714	385	924	109	839	284	687	39	429	7	421	207	167	753	529	231	728	590	423	964
704	252	974	399	525	825	726	599	518	755	219	447	951	409	853	293	270	781	26	532	523	69	122	316	457	158	866
298	197	976	660	659	5	289	535	907	488	617	283	650	60	863	201	146	968	580	325	99	410	120	301	684	651	742
198	785	757	902	910	557	602	601	827	677	560	803	266	43	811	730	612	112	32	40	455	367	521	758	66	920	811
690	390	120	652	942	669	242	340	798	116	887	716	312	109	662	568	174	483	34	226	131	376	893	498	833	789	867
195	453	70	81	182	415	688	667	555	46	872	834	536	117	422	432	527	32	547	981	276	640	961	739	160	14	657
526	250	197	12	286	276	149	942	138	415	318	143	181	904	200	350	135	417	614	726	655	728	837	536	587	377	540
467	419	39	820	523	633	810	121	163	459	920	130	778	595	378	757	432	198	133	181	492	715	220	738	710	749	294
787	485	769	976	64	491	850	491	816	418	1	642	670	800	864	205	587	586	801	470	294	357	31	772	315	312	571
193	175	257	739	942	985	397	168	566	453	980	133	873	655	801	789	44	146	803	629	211	751	565	834	531	938	470
738	37	465	549	447	812	956	248	163	479	529	685	144	710	690	864	418	87	609	595	926	723	444	913	299	2	708
561	30	486	115	364	916	658	892	60	499	782	747	936	517	687	881	469	245	870	122	539	249	695	32	180	544	606
208	938	333	563	383	927	710	611	31	244	394	74	478	2	428	490	743	374	476	656	772	2	36	595	632	243	654
469	308	169	413	70	759	278	665	867	42	659	585	634	46	599	878	28	602	457	523	337	595	411	593	357	139	563
383	711	896	804	280	42	761	893	374	240	272	522	837	452	876	784	692	475	273	379	548	711	288	436	864	698	869
230	396	922	911	501	900	480	889	490	527	300	183	548	30	118	434	329	589	458	156	662	37	336	725	117	371	521
265	409	693	948	20	468	681	693	400	380	682	139	366	978	269	47	120	481	54	541	877	314	223	90	379	222	908
388	539	652	231	483	948	835	925	58	333	427	798	562	114	744	60	297	582	420	237	385	771	967	886	828	969	462
755	99	429	900	41	730	640	627	627	220	334	100	850	897	598	932	491	628	601	449	662	577	473	599	940	160	917
906	491	913	504	419	956	59	980	603	923	198	117	52	978	535	322	626	791	850	346	121	114	71	196	963	945	960
523	31	32	489	488	745	902	150	627	2	825	384	612	441	714	82	63	312	197	927	510	536	981	180	262	84	14
10	552	603	220	492	568	748	717	614	132	619	811	85	411	455	927	843	713	833	825	899	217	61	663	118	297	983
432	864	983	360	749	508	813	901	22	757	489	923	471	454	39	803	707	726	242	780	31	214	16	453	494	801	322
491	967	297	800	381	618	488	692	359	573	477	883	348	311	887	234	679	393	193	652	673	476	496	344	978	417	369
519	365	955	907	597	888	567	148	400	723	10	975	279	206	587	376	635	888	767	384	382	61	624	960	453	446	596
544	795	737	467	774	471	778	946	259	50	704	391	369	572	468	124	436	305	86	698	115	865	218	780	453	202	416
502	158	181	675	322	146	909	464	848	594	493	513	947	510	665	200	587	631	745	705	974	954	41	835	641	396	669
884	527	210	102	806	151	761	645	100	468	78	193	440	535	883	348	189	55	225	197	977	571	581	323	273	639	160
901	91	4	725	584	396	501	574	707	763	255	884	890	680	234	5	309	566	369	309	726	54	10	502	347	460	930
88	281	572	818	741	619	521	538	502	297	736	777	485	697	538	232	361	540	145	47	407	321	212	461	485	409	452
883	557	370	209	942	772	74	32	551	129	751	820	856	973	764	171	950	336	397	200	543	654	441	19	17	63	977
827	588	863	744	347	435	20	610	495	328	874	344	767	392	17	272	432	968	235	668	508	50	662	384	46	922	211
961	519	248	80	776	247	735	393	17	747	102	969	903	100	414	456	67	795	665	9	715	865	403	372	641	642	285
519	737	692	765	726	741	176	70	521	731	731	389	50	929	154	526	579	794	60	971	946	834	580	924	540	787	575
332	428	815	177	515	15	647	475	195	973	570	841	522	804	26	402	47	261	795	366	566	171	693	318	904	378	974
182	299	918	234	561	961	134	864	829	341	262	792	588	540	52	441	699	477	283	442	524	492	871	340	580	560	241

Αλλά ακόμη και στην περίπτωση αυτή, ο άνθρωπος, μπορεί να οργανώσει μια στρατηγική που να στηρίζεται στο γεγονός ότι είναι σε θέση να συλλαμβάνει ταυτόχρονα και να επεξεργάζεται παραστάσεις διεσπαρμένες στο χώρο, ενώ ένας ΗΥ δε μπορεί.

Πριν από αυτό όμως πρέπει να επισημανθεί το γεγονός ότι στο παραπάνω παράδειγμα, οι αριθμοί είναι όλοι γνωστοί και διαθέσιμοι, ενώ στο πρόβλημα που θέσαμε δεν είναι κατ' ανάγκη.

Μια σειρά αριθμών μπορεί λοιπόν να σημαίνει είτε ένα συγκεκριμένο πλήθος αριθμών, είτε ότι έχουμε ένα απροσδιόριστο πλήθος αριθμών – στην περίπτωση αυτή όμως θα είναι γνωστό το τελευταίο στοιχείο της σειράς, δηλαδή των αριθμών (έστω το 0).

Στην πρώτη περίπτωση απαιτείται λοιπόν η χρήση ενός μετρητή ενώ στη όχι.

Παρατίθενται και οι δυο λύσεις:

```
begin
read n;
if n > 0 then
  begin
  count:=1;
  read min;
  while count < > n+1 do
    begin
    read x;
    if min > x then min:=x;
    count:=count + 1;
    end;
  write min;
  end;
end.
```

```
begin
read min;
if min <> 0 then
  begin
  read x;
  while x <> 0 do
    begin
    if  min > x then min:=x;
    read x;
    end;
  write min;
  end;
end.
```

Και στις δυο λύσεις ελέγχεται αρχικά μήπως το πρώτο από τα δεδομένα αποτελεί ήδη το σημείο τέλους, δηλαδή ελέγχεται μήπως η διαδοχή είναι κενή. Αυτό βέβαια μπορεί να φανεί χωρίς νόημα στα μάτια των μαθητών οι οποίοι μπορούν να αναρωτηθούν τι νόημα έχει να αναζητήσει κανείς τον ελάχιστο από μια σειρά ανύπαρκτων αριθμών!

Στη πραγματικότητα ο έλεγχος αυτός εξασφαλίζει την ορθότητα του προγράμματος για όλες τις διαδοχές αριθμών – ακόμη και για την κενή διαδοχή – αλλά έχει και πρακτικό νόημα. Καθώς ο έλεγχος του ελαχίστου μπορεί να είναι ένα τμήμα ενός άλλου προγράμματος, δεν επιθυμούμε να υπάρχουν περιπτώσεις που δεν καλύπτονται από το πρόγραμμα. Για παράδειγμα, έστω ένα πρόγραμμα που ελέγχει κάθε χρόνο το μικρότερο ύψος από τους σπουδαστές κάθε νομού, ανά Πανεπιστημιακή Σχολή. Είναι όμως δυνατόν μια χρονιά να μην εισαχθούν σπουδαστές στη Νομική Σχολή από έναν νομό. Τότε προφανώς το αντίστοιχο σύνολο των σπουδαστών από το νομό αυτό θα είναι κενό και το πρόγραμμα θα επιχειρήσει να προσδιορίσει τον ελάχιστο από το κενό αυτό σύνολο. Στις περιπτώσεις λοιπόν αυτές, επιθυμούμε να έχει προβλεφθεί και η περίπτωση του κενού συνόλου – και για το λόγο αυτό πάντοτε ελέγχουμε αρχικά μήπως η διαδοχή μας είναι κενή.

20. Να γραφεί ένα πρόγραμμα, έτσι ώστε από μια σειρά άνισων αριθμών να προσδιοριστεί ο ελάχιστος και ο μέγιστος.

Το πρόβλημα αυτό είναι βέβαια ανάλογο προς το προηγούμενο. Το μόνο λεπτό σημείο είναι ο προσδιορισμός του ελαχίστου και του μεγίστου. Αν η διαδοχή είναι 4,4,4, τότε ποιος είναι ο μέγιστος και ο ελάχιστος; Για τη λύση υιοθετούμε την (μαθηματικώς ορθή) απάντηση ότι το 4 είναι και ο μέγιστος και ο ελάχιστος, αλλά αυτό μπορεί να μη συμφωνεί με τις διαισθητικές

αντιλήψεις του είδους «ο μέγιστος πρέπει να είναι μεγαλύτερος από τον ελάχιστο». Ο διδάσκων θα πρέπει να το επισημάνει στους μαθητές δείχνοντας τους την ανάγκη για επακριβή ορισμό των όρων που υπάρχουν σε ένα πρόβλημα.

```
begin
read min;
max:=min;
if min <> 0 then
  begin
  read x;
  while x <> 0 do
    begin
    if min > x then min:=x;
    if x > max then max:=x;
    read x;
    end;
  end;
write min;
write max;
end.
```

21. Να γραφεί ένα πρόγραμμα, έτσι ώστε σε μια σειρά αριθμών να υπολογιστεί το άθροισμα τους.

Στα εξής θα θεωρούμε ότι η έκφραση σε μια σειρά αριθμών θα αντιστοιχεί σε μια – ενδεχομένως κενή –διαδοχή αγνώστου πλήθους αριθμούς με γνωστό τον τελευταίο.

Στο συγκεκριμένο πρόβλημα δε διευκρινίζεται αν ο τελευταίος αριθμός θα πρέπει να προστεθεί ή όχι – και πάλι η επιλογή δεν υπαγορεύεται από τη λογική της λύσης αλλά από τη χρήση του προγράμματος. Θα υποθέσουμε ότι ο τελευταίος αριθμός είναι ο 999 και θα παρουσιάσουμε δυο λύσεις: στη μια ο 999 θα προστίθεται στο άθροισμα, ενώ στην άλλη όχι:

```
begin
read x;
sum:=x;
while x <> 999 do
  begin
  read x;
  sum:=sum+x;
  end;
write sum;
end.
```

```
begin
read x;
sum:=0;
while x <> 999 do
  begin
  sum:=sum+x;
  read x;
  end;
write sum;
end.
```

Ο διδάσκων θα πρέπει να υπογραμμίσει τις λεπτές διαφορές στον κώδικα των δυο προγραμμάτων που προέρχονται και μόνο από την επιλογή της πρόσθεσης του 999 στο τελικό άθροισμα.

22. Να γραφεί ένα πρόγραμμα, για τον υπολογισμό του μέσου όρου μιας σειράς αριθμών (μόνο το ακέραιο πηλίκο).

Το πρόβλημα είναι τελείως ανάλογο προς το προηγούμενο, αλλά καθώς το πλήθος των αριθμών είναι άγνωστο, πρέπει να υπάρχει και ένας μετρητής ώστε να χρησιμοποιηθεί στο τέλος. Αν η διαδοχή είναι κενή, ο μέσος όρος δε μπορεί να υπολογιστεί και η διαίρεση στον τύπο του μέσου όρους δεν πραγματοποιείται.

```
begin
read x;
counter:=1;
sum:=x;
while x <> 999 do
  begin
  read x;
  sum:=sum+x;
  counter:=counter + 1;
  end;
mo:=sum / counter
write mo;
end.
```

```
begin
read x;
counter:=0;
sum:=0;
while x <> 999 do
  begin
  sum:=sum+x;
  counter:=counter + 1;
  read x;
  end;
if counter > 0 then
  begin
  mo:=sum/counter;
  write mo;
  end;
end.
```

23. Να υπολογιστεί το άθροισμα 1+2+3+…+300.

Υποθέτουμε ότι οι μαθητές της Α' Λυκείου δε γνωρίζουν τους τύπους των αριθμητικών προόδων και άρα δε θα επιλύσουν το πρόβλημα χρησιμοποιώντας τύπους.

Η βασική δυσκολία στην επίλυση του προβλήματος αυτού έγκειται, κατά τη γνώμη μας, στη μεγάλη απόσταση που υφίσταται ανάμεσα στη «φυσική» λύση με το χέρι και στις απαιτήσεις μίας λύσης με τη βοήθεια ενός ΗΥ. Επιπλέον είναι απαραίτητη η ανακάλυψη κάποιων αναλλοίωτων σχέσεων οι οποίες αποτελούν τη λύση του προβλήματος και οι σχέσεις αυτές εισάγουν μια μεταβλητή η οποία εκ πρώτης όψεως δε φαίνεται να είναι απαραίτητη: του αθροιστή.

Έτσι, οι πρώτες απόπειρες των μαθητών είναι καταδικασμένες σε αποτυχία:

οι μαθητές σκέφτονται *παίρνουμε έναν αριθμό και ύστερα τον επόμενο... δηλαδή x + x+1...*

και προφανώς η στρατηγική αυτή δεν οδηγεί στη λύση. Η πληροφορική λύση χρησιμοποιεί έναν αθροιστή (μια μεταβλητή η οποία δεν είναι προφανής από την εκφώνηση του προβλήματος) ο οποίος αθροίζει τις διαδοχικές τιμές ενός μετρητή (ο οποίος μετρητής παίζει το διπλό ρόλο της μεταβλητής που ελέγχει την επαναληπτική διαδικασία αλλά και της «γεννήτριας» των φυσικών που προστίθενται στον αθροιστή).

```
begin
counter:=0;
sum:=0;
while 301 > counter do
  begin
  counter:=counter+1;
  sum:=sum+counter;
  end;
write sum;
end.
```