Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση

Τόμ. 1 (2005)

3ο Συνέδριο Σύρου στις ΤΠΕ

Μετατροπή ενός αλγορίθμου με εντολές άλματος σε δομημένο αλγόριθμο: Μια διδακτική προσέγγιση

Ευριπίδης Βραχνός

Βιβλιογραφική αναφορά:

Βραχνός Ε. (2024). Μετατροπή ενός αλγορίθμου με εντολές άλματος σε δομημένο αλγόριθμο: Μια διδακτική προσέγγιση. *Συνέδρια της Ελληνικής Επιστημονικής Ένωσης Τεχνολογιών Πληροφορίας & Επικοινωνιών στην Εκπαίδευση*, *1*, 195–202. ανακτήθηκε από https://eproceedings.epublishing.ekt.gr/index.php/cetpe/article/view/6209

ΜΕΤΑΤΡΟΠΗ ΕΝΟΣ ΑΛΓΟΡΙΘΜΟΥ ΜΕ ΕΝΤΟΛΕΣ ΆΛΜΑΤΟΣ ΣΕ ΔΟΜΗΜΕΝΟ ΑΛΓΟΡΙΘΜΟ: ΜΙΑ ΔΙΔΑΚΤΙΚΗ ΠΡΟΣΕΓΓΙΣΗ

Βραχνός Ευριπίδης
Καθηγητής Πληροφορικής
ΤΕΕ Μήλου
evrachnos@gmail.com
http://www.db-net.aueb.gr/evry/

ΠΕΡΙΛΗΨΗ
Η εντολή άλματος goto έχει εδώ και πολλά χρόνια τεθεί στο περιθώριο του προγραμματισμού τόσο σε εκπαιδευτικό όσο και σε επαγγελματικό επίπεδο. Ωστόσο υπάρχουν κάποια διδακτικά οφέλη που μπορούμε να αποκομίσουμε από αυτήν. Η εργασία αυτή ασχολείται με την μετατροπή αλγορίθμων με εντολή άλματος σε δομημένα ισοδύναμά τους. Θα δείξουμε ότι η μετατροπή αυτή έχει σημαντική εκπαιδευτική αξία, αφού δίνει στον καθηγητή πληροφορικής ένα ακόμη εργαλείο για την κατανόηση από τον μαθητή της ροής του ελέγχου σε ένα πρόγραμμα. Επίσης ο μαθητής θα καταλάβει την ανάγκη για τη χρήση δομών επιλογής και επανάληψης όπως επίσης και τη σημασία του δομημένου προγραμματισμού. Για το σκοπό αυτό κάνουμε χρήση της διαγραμματικής αναπαράστασης του αλγορίθμου, σχεδιάζοντας το αντίστοιχο διάγραμμα ροής ώστε να γίνει φανερή η αντιστοιχία της εντολής άλματος με κάποια από τις δομές επιλογής ή επανάληψης. Με κατάλληλη αντικατάσταση προκύπτει ο δομημένος αλγόριθμος.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: *εντολή άλματος, goto, διάγραμμα ροής, δομημένος προγραμματισμός , ανάπτυξη εφαρμογών*

ΕΙΣΑΓΩΓΗ

Η εντολή άλματος goto έχει ως αποτέλεσμα την αλλαγή της ροής του προγράμματος, δηλαδή τη μεταφορά του ελέγχου σε οποιοδήποτε σημείο του προγράμματος. Ουσιαστικά παραβιάζει τη δομή της ακολουθίας αφού ο έλεγχος μπορεί να μεταβεί σε οποιοδήποτε σημείο του προγράμματος κάνοντας πολύ δύσκολη την κατανόηση και τη συντήρησή του.

Η εντολή άλματος JMP είναι η αντίστοιχη εντολή στις συμβολικές γλώσσες (assembly languages) που υλοποιεί την αλλαγή στη ροή του ελέγχου. Δηλαδή όλες οι δομές επιλογής και επανάληψης που ξέρουμε όταν μετατρέπονται σε γλώσσα μηχανής υλοποιούνται με ισοδύναμες εντολές άλματος. Έτσι η εντολή αυτή διαδόθηκε και στις πρώτες γλώσσες προγραμματισμού υψηλού επιπέδου όπως οι Fortran, COBOL και έγινε δημοφιλής μέσα από τη διάδοση της Basic η οποία έγινε συνώνυμο της εντολής άλματος και γενικότερα του μη δομημένου προγραμματισμού. Μάλιστα παρ'όλα τα εμφανή προβλήματα της εντολής, πολλοί προγραμματιστές υποστήριζαν τη χρήση της με επιχείρημα ότι στα σωστά "χέρια" μπορεί να αποδειχθεί ένα πολύ σημαντικό εργαλείο για τον προγραμματιστή. Όσο όμως αυξανόταν ο όγκος των προγραμμάτων στα οποία γινόταν αλόγιστη χρήση της εντολής goto, η εκσφαλμάτωση, συντήρηση και επέκταση των προγραμμάτων αυτών γινόταν όλο και πιο δύσκολη και σε κάποιες περιπτώσεις πρακτικά αδύνατη.

Η πρώτη αντίδραση της επιστημονικής κοινότητας ήταν η θεωρητική δουλειά των Bohm και Jacopini (Bohm 1966), που αποτελούσε βελτίωση μιας δημοσίευσης που είχαν ήδη κάνει το 1964 σε ένα συνέδριο στο Ισραήλ. Ωστόσο η δουλειά αυτή έτυχε αναγνώρισης από όλο τον κόσμο κυρίως μετά τη φημισμένη πρόταση *Go to statement considered harmful* του E.W. Dijkstra

(Dijkstra 1968). Ο Dijkstra δεν είχε κάποια θεωρητική απόδειξη όπως οι Bohm και Jacopini, είχε όμως πειστικά επιχειρήματα ότι η αλόγιστη χρήση της εντολής goto μπορεί να δημιουργήσει μεγάλα προβλήματα τα οποία αυξάνονται όσο αυξάνεται το μέγεθος του προγράμματος. Τη δεκαετία του 70' με την εξάπλωση γλωσσών που προήγαγαν τον δομημένο προγραμματισμό όπως η ALGOL και αργότερα η PASCAL, άρχισε να περιορίζεται σημαντικά η χρήση της εντολής goto. Σήμερα εντολές άλματος δεν χρησιμοποιούνται σχεδόν καθόλου εκτός από εξαιρετικές περιπτώσεις και κυρίως για λόγους συμβατότητας. Στη Java μάλιστα δεν υπάρχει η εντολή goto αλλά τα υποκατάστατά της break και continue τα οποία δεν προκαλούν τα ίδια προβλήματα, όπως επίσης και οι εξαιρέσεις (exceptions) για τον χειρισμό σφαλμάτων.

Η εργασία αυτή πραγματεύεται έναν τρόπο αφαίρεσης της εντολής άλματος από ένα τμήμα αλγορίθμου και τη μετατροπή του στο δομημένο ισοδύναμό του, στα πλαίσια του μαθήματος "Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον" της τεχνολογικής κατεύθυνσης της Γ' Λυκείου. Στο σχολικό βιβλίο δίνεται ο ορισμός του δομημένου προγραμματισμού και γίνεται μια αναφορά στην εντολή goto. Στο τετράδιο μαθητή όμως υπάρχουν ασκήσεις όπου δίνεται ένα τμήμα μη δομημένου προγράμματος και ζητείται από τον μαθητή να σχεδιάσει αλγόριθμο με τις αρχές του δομημένου προγραμματισμού που να εκτελεί τις ίδιες λειτουργίες. Εμείς θα προτείνουμε έναν τρόπο διδασκαλίας για την απαλοιφή της εντολής άλματος. Στόχος μας δεν είναι η σωστή επίλυση αντίστοιχων ασκήσεων, αλλά η κατανόηση της ροής του ελέγχου σε ένα πρόγραμμα από τον μαθητή.

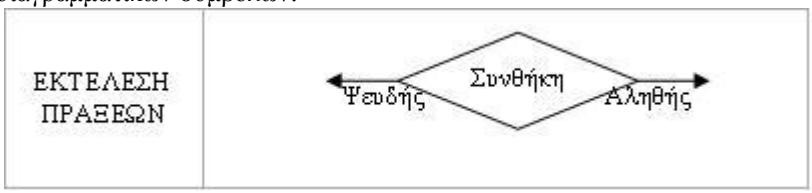
Όπως θα δούμε παρακάτω μια εντολή goto μπορεί να μετασχηματιστεί σε δομή επιλογής ή δομή επανάληψης. Με αυτόν τον τρόπο ο μαθητής θα καταλάβει καλύτερα πως λειτουργούν οι δομές επανάληψης και επιλογής και πως αυτές μετατρέπονται σε εντολές άλματος JMP όταν ο μεταγλωττιστής μετατρέπει ένα πρόγραμμα από γλώσσα υψηλού επιπέδου σε γλώσσα μηχανής.

Η ΔΙΑΓΡΑΜΜΑΤΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ

Υπάρχουν πολλές τεχνικές για την μετατροπή ενός αδόμητου τμήματος αλγορίθμου σε δομημένο. Οι τεχνικές αυτές χρησιμοποιούνται για τον σχεδιασμό των γνωστών *decompilers*. Αυτά τα προγράμματα εκτελούν την αντίστροφη εργασία από αυτήν ενός μεταγλωττιστή. Δηλαδή δέχονται ως είσοδο ένα πρόγραμμα σε γλώσσα μηχανής και το μετατρέπουν σε μια γλώσσα υψηλού επιπέδου. Κατά τη μετατροπή αυτή θα πρέπει να απαλείψουν όλες τις εντολές άλματος και να τις αντικαταστήσουν με τα δομημένα ισοδύναμά τους. Αντίστοιχες τεχνικές χρησιμοποιούνται επίσης και για την παραλληλοποίηση προγραμμάτων έτσι ώστε αυτά να μπορούν να εκτελεστούν από περισσότερους από έναν επεξεργαστές. Όταν ένα πρόγραμμα είναι σχεδιασμένο με τις αρχές του δομημένου προγραμματισμού είναι πολύ πιο εύκολο να χωριστεί σε τμήματα ανεξάρτητα μεταξύ τους τα οποία θα μπορούν να εκτελεστούν παράλληλα από διαφορετικούς επεξεργαστές.

Το πρόβλημα με όλες αυτές τις τεχνικές είναι ότι δεν μπορούν να χρησιμοποιηθούν σε ένα μάθημα όπως αυτό της Γ' Λυκείου λόγω της πολυπλοκότητάς τους, που θα τις κάνει δυσνόητες για τους μαθητές. Συνήθως η γεωμετρική μορφή ενός προβλήματος υπερέχει αισθητά από την αλγεβρική του όσον αφορά τη διδακτική τους αξία, δηλαδή την ευκολία κατανόησής τους από τον μαθητή. Η γεωμετρική αναπαράσταση για το πρόβλημα που εξετάζουμε είναι η διαγραμματική αναπαράσταση του αλγορίθμου. Θα χρησιμοποιήσουμε τα *διαγράμματα ροής*, μια και είναι η διαγραμματική αναπαράσταση που χρησιμοποιείται στο σχολικό βιβλίο (Βακάλη 1999). Μάλιστα, αυτή η αναπαράσταση ταιριάζει απόλυτα με το πρόβλημα που εξετάζουμε διότι είναι σχεδιασμένη για να δείχνει αυτό που είναι το ζητούμενο για μας, δηλαδή τη ροή του ελέγχου σε ένα πρόγραμμα.

Ο Scanlan στην εργασία του (Scanlan 1989), μετά από εκτενή σύγκριση της αναπαράστασης ενός αλγορίθμου με διαγράμματα ροής και με ψευδοκώδικα, φτάνει στο συμπέρασμα ότι τα διαγράμματα ροής έχουν μεγαλύτερη διδακτική αξία. Ειδικά για την περίπτωση της κατανόησης της ροής του ελέγχου ενός προγράμματος είναι ιδανικά. Στη συνέχεια θα μελετήσουμε τμήματα αλγορίθμων που δεν θα έχουν λειτουργίες εισόδου-εξόδου. Έτσι θα κάνουμε χρήση μόνο των δυο παρακάτω διαγραμματικών συμβόλων:



Σχήμα 1. *Το ορθογώνιο δηλώνει την εκτέλεση μιας ή περισσότερων πράξεων και ο ρόμβος μια εντολή διακλάδωσης ανάλογα με την τιμή της Συνθήκης.*

Να σημειώσουμε ότι οι αλγοριθμικές δομές λόγω της σκοπιάς από την οποία τις εξετάζουμε, είναι οι ίδιες είτε αναφερόμαστε σε αλγόριθμο είτε σε πρόγραμμα. Για αυτό οι όροι αλγόριθμος και πρόγραμμα θα χρησιμοποιούνται σε αυτό το κείμενο χωρίς διάκριση. Επίσης όταν λέμε ότι δυο αλγόριθμοι, προγράμματα ή διαγράμματα ροής είναι *ισοδύναμα* θα εννοούμε ότι για όλες τις δυνατές εισόδους, όλες οι εκτελέσεις επιστρέφουν το ίδιο αποτέλεσμα.

Τέλος η σημειολογία που χρησιμοποιούμε για τα διαγράμματα ροής έχει κάποιες μικροδιαφορές με αυτή του βιβλίου, όμως η συζήτηση με τους μαθητές έδειξε ότι τους βοηθάει καλύτερα να καταλάβουν τη λειτουργία της εντολής άλματος και την αντιστοιχία της με τις δομές επιλογής και επανάληψης κατά περίπτωση.

ΑΝΑΛΥΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Ας δούμε ένα παράδειγμα μη δομημένου προγράμματος:

```
Εντολή_1
GOTO Ετικέτα
Εντολή_2
Εντολή_3
Ετικέτα: Εντολή_4
```

Η εκτέλεση της εντολής *GOTO* έχει σαν αποτέλεσμα τη μεταφορά του ελέγχου στην *Ετικέτα* και την εκτέλεση της εντολής *Εντολή_4*. Δηλαδή οι εντολές *Εντολή_2*, *Εντολή_3* παρακάμπτονται και δεν εκτελούνται ποτέ. Άρα το αντίστοιχο δομημένο πρόγραμμα θα περιέχει μόνο τις εντολές που θα εκτελεστούν με τη συγκεκριμένη σειρά, δηλαδή τις εντολές 1,4.

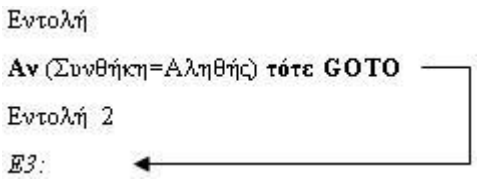
Υπάρχουν δυο περιπτώσεις: (1) η εντολή άλματος και η ετικέτα προορισμού να βρίσκονται στην ίδια ομάδα(μπλοκ) εντολών και (2) να βρίσκονται σε διαφορετική ομάδα εντολών.

Όταν η εντολή άλματος και η ετικέτα είναι και οι δυο στην ίδια ομάδα εντολών η απάλειψη της εντολής άλματος είναι πολύ απλή και για αυτό θα ασχοληθούμε κυρίως με την περίπτωση αυτή. Ωστόσο θα δώσουμε στη συνέχεια και παραδείγματα της δεύτερης περίπτωσης.

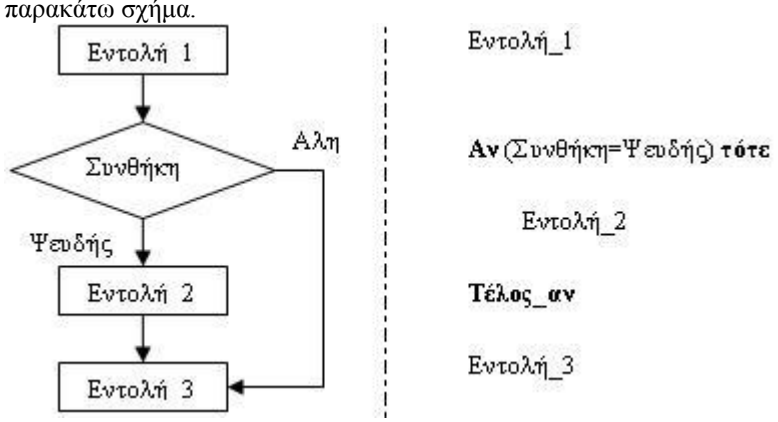
Στην πρώτη περίπτωση υπάρχουν δυο πιθανότητες: η εντολή άλματος (α) να οδηγεί σε επόμενη εντολή (β) να οδηγεί σε προηγούμενη εντολή.

ΔΟΜΗ ΕΠΙΛΟΓΗΣ

Ας ξεκινήσουμε με ένα παράδειγμα όπου η εντολή άλματος οδηγεί σε επόμενη εντολή.



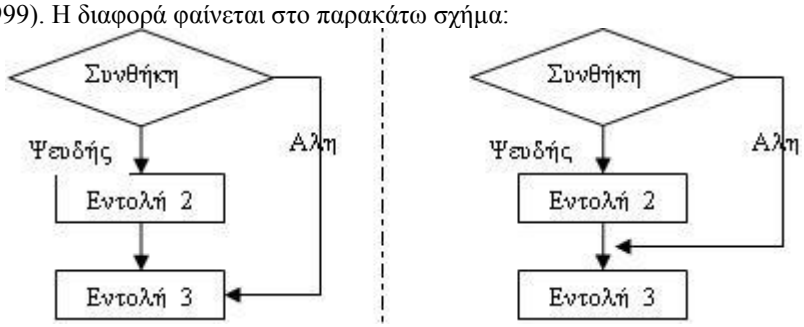
Στην πρώτη επαφή που θα έχει ο μαθητής με μια εντολή άλματος ο σχεδιασμός ένος βέλους που οδηγεί στην εντολή που θα εκτελεστεί όπως φαίνεται παραπάνω ίσως θα ήταν καλή ιδέα. Θα τον βοηθήσει να σχεδιάσει το αντίστοιχο διάγραμμα ροής, αφού καταλάβει ότι μια εντολή άλματος ισοδυναμεί με μεταφορά της ροής του ελέγχου σε άλλο σημείο του προγράμματος. Στη συνέχεια η κατασκευή του αντίστοιχου διαγράμματος ροής δεν είναι δύσκολη υπόθεση. Αφού ο μαθητής κατασκευάσει το διάγραμμα ροής του ζητάμε στη συνέχεια να το μετατρέψει σε τμήμα αλγορίθμου χωρίς να χρησιμοποιήσει την εντολή goto. Η μετατροπή είναι εξαιρετικά απλή και φαίνεται στο παρακάτω σχήμα.



Σχήμα 2. *Μετά την κατασκευή του διαγράμματος ροής (αριστερά) η μετατροπή του σε τμήμα αλγορίθμου γίνεται κατ'ευθείαν (δεξιά) με χρήση της δομής Αν...Τότε.*

Μέσα από την παραπάνω διαδικασία είναι πολύ πιο εύκολο για τον μαθητή να κατανοήσει αλλά και να ανακαλύψει μόνος του τον τρόπο λειτουργίας της εντολής επιλογής *Αν...τότε* και του τρόπου με τον οποίο αλλάζει τη σειρά της εκτέλεσης των εντολών στο πρόγραμμα. Η εντολή goto είναι απλά το μέσο.

Στο παραπάνω παράδειγμα θα πρέπει να σημειώσουμε ότι το βέλος καταλήγει πάνω στην Εντολή 3 και όχι στη γραμμή ελέγχου που οδηγεί σε αυτήν όπως συμβαίνει στο σχολικό βιβλίο (Βακάλη 1999). Η διαφορά φαίνεται στο παρακάτω σχήμα:



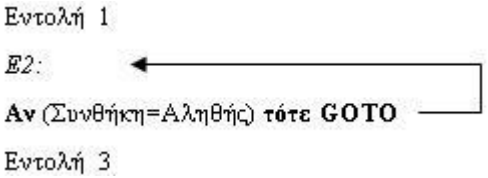
Σχήμα 3. *Αριστερά φαίνεται ο τρόπος σχεδιασμού που χρησιμοποιούμε σε αυτήν την εργασία και δεξιά ο τρόπος του σχολικού βιβλίου*

Για να μη δημιουργηθεί κάποια παρεξήγηση να πούμε ότι ο τρόπος του σχολικού βιβλίου είναι ο σωστός και ο ευρέως αποδεκτός τρόπος. Ο λόγος που κάναμε αυτή τη μικρή αλλαγή ήταν η επιμονή των μαθητών να μεταφράζουν την εντολή άλματος με αυτόν τον τρόπο στο διάγραμμα ροής και είναι απόλυτα λογικό αφού το σημείο προορισμού είναι μια άλλη εντολή.

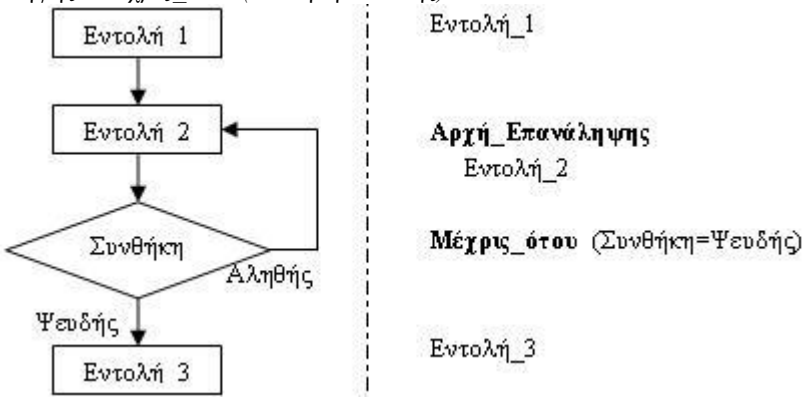
Ας δούμε τώρα και ένα παράδειγμα για την περίπτωση που η εντολή άλματος οδηγεί σε προηγούμενη εντολή.

ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ

Ας δούμε τώρα και ένα παράδειγμα για την περίπτωση που η εντολή άλματος οδηγεί σε προηγούμενη εντολή.



Σε αυτή την περίπτωση οι περισσότεροι μαθητές θα καταλάβουν ότι πρόκειται για εντολή επανάληψης, όμως δεν θα μπορέσουν να διακρίνουν εύκολα το είδος της επανάληψης. Η μετατροπή του προηγούμενου τμήματος αλγορίθμου σε ισοδύναμο διάγραμμα ροής θα δώσει τη λύση απλά και εύκολα όπως φαίνεται στο παρακάτω σχήμα. Από το διάγραμμα ροής είναι πλέον φανερό ότι η δομή επανάληψης για τη συγκεκριμένη περίπτωση είναι μια δομή της μορφής *Επανάλαβε...Όσο (Συνθήκη=Αληθής)*(υπάρχει στις γλώσσες *C++* και *Java* ως *do{…}while (Συνθήκη)*). Η αντίστοιχη δομή επανάληψης που προτείνει το σχολικό βιβλίο είναι η *Αρχή_Επανάληψης...Μέχρις_ότου (Συνθήκη=Ψευδής)*.



Σχήμα 4. *Μετά την κατασκευή του διαγράμματος ροής (αριστερά) η μετατροπή του σε τμήμα αλγορίθμου γίνεται κατ'ευθείαν (δεξιά) με χρήση της δομής Αρχή_επανάληψης...Μέχρις_ότου.*

Αυτή η αλλαγή της συνθήκης σε ψευδή στη δομή *Μέχρις_ότου* μπορεί να προκαλέσει σύγχυση στους μαθητές. Αν υποθέσουμε ότι οι περισσότεροι μαθητές δεν είχαν διδαχθεί καμία από τις δομές επανάληψης του σχολικού βιβλίου (*Όσο...επανάλαβε*, *Αρχή_επανάληψης...Μέχρις_ότου*) το πιθανότερο είναι ότι θα χρησιμοποιούσαν τη δομή επανάληψης *Επανάλαβε...Όσο (Συνθήκη=Αληθής)*. Η μετατροπή όμως αυτής της δομής σε

Αρχή_επανάληψης...Μέχρις_ότου επιβάλλει τη λογική άρνηση της συνθήκης όπως φαίνεται και στο παραπάνω σχήμα.

Η ΜΕΘΟΔΟΛΟΓΙΑ

Αν θέλουμε λοιπόν να συνοψίσουμε τα βήματα που θα πρέπει να ακολουθήσει ο μαθητής για να απαλείψει μια εντολή άλματος για τις παραπάνω απλές περιπτώσεις έχουμε:

1. Συνδέουμε με μια γραμμή-βέλος την εντολή άλματος με την εντολή στην οποία οδηγεί.
2. Σχεδιάζουμε το αντίστοιχο διάγραμμα ροής.
3. Διακρίνουμε αν πρόκειται για δομή επιλογής ή επανάληψης από την κατεύθυνση της ροής του ελέγχου.
4. Γράφουμε το αντίστοιχο τμήμα αλγορίθμου.

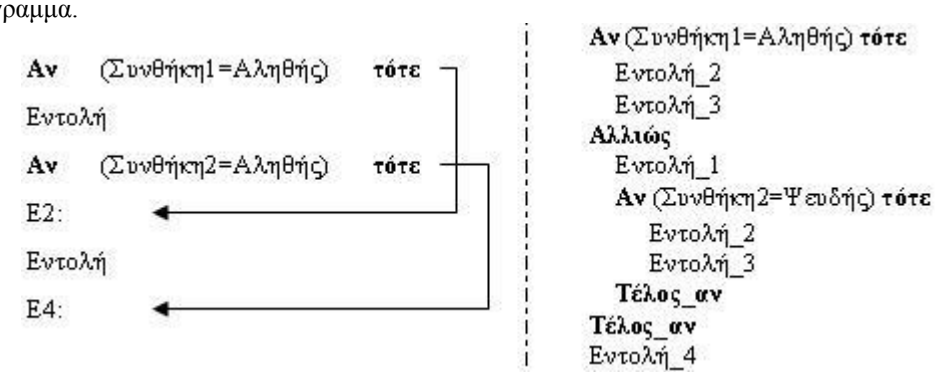
Τα βήματα αυτά δεν αποτελούν πανάκεια ούτε λύνουν με εξίσου εύκολο τρόπο όλα τα προβλήματα. Όμως πιστεύουμε ότι είναι ένας πολύ ομαλός και κατανοητός τρόπος για μια πρώτη επαφή του μαθητή με την εντολή άλματος *GOTO* και την απαλοιφή της από ένα πρόγραμμα έτσι ώστε αυτό να πληροί τις βασικές αρχές του δομημένου προγραμματισμού. Ένας άλλος τρόπος επίλυσης τέτοιων ασκήσεων είναι η καταγραφή όλων των σεναρίων εκτέλεσης για κάθε περίπτωση και η δημιουργία από αυτά, του δομημένου προγράμματος. Κάτι τέτοιο όμως δεν ενδείκνυται, κατά τη γνώμη μας, για μια πρώτη επαφή με το αντικείμενο.

ΠΕΠΛΕΓΜΕΝΕΣ ΕΝΤΟΛΕΣ ΑΛΜΑΤΟΣ

Στην προηγούμενη ενότητα αναφερθήκαμε σε απλές περιπτώσεις όπου η εντολή άλματος και η εντολή προορισμού βρίσκονται στην ίδια ομάδα εντολών. Ο σκοπός ήταν να δει ο μαθητής ένα παράδειγμα μη δομημένου προγραμματισμού, έτσι ώστε να κατανοήσει καλύτερα τις βασικές αρχές του και όχι να μπορεί να μετατρέπει πολύπλοκα αδόμητα προγράμματα σε δομημένα ισοδύναμά τους. Ωστόσο στο τετράδιο του μαθητή υπάρχουν δυο ασκήσεις στις οποίες η εντολή άλματος οδηγεί σε διαφορετικό ή και εμφωλευμένο μπλοκ εντολών. Για λόγους πληρότητας θα περιγράψουμε στη συνέχεια δυο τέτοια παραδείγματα αδόμητων προγραμμάτων όπως περιγράφονται στο (Zhang et. al. 2004). Στην εργασία αυτή, ο αναγνώστης μπορεί να δει και τις σχετικές αποδείξεις σχετικά την ορθότητα του μετασχηματισμού.

ΠΕΠΛΕΓΜΕΝΗ ΔΟΜΗ ΕΠΙΛΟΓΗΣ

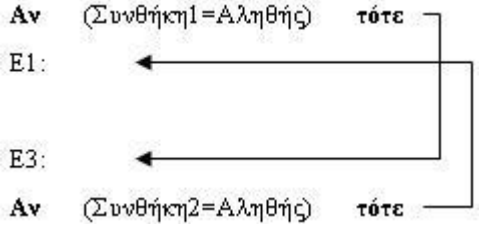
Η περίπτωση που θα εξετάσουμε εδώ είναι η απλούστερη από τις τρεις. Εδώ ο μαθητής θα μπορούσε να αναλύσει όλα τα πιθανά σενάρια εκτέλεσης που προκύπτουν από τους συνδυασμός των τιμών των δυο συνθηκών. Έτσι θα μπορούσε να προκύψει το παρακάτω δομημένο πρόγραμμα.



Σχήμα 5. *Αριστερά δίνεται ένα πρόγραμμα με πεπλεγμένες εντολές άλματος και δεξιά το δομημένο ισοδύναμό του*

ΠΕΠΛΕΓΜΕΝΗ ΔΟΜΗ ΕΠΑΝΑΛΗΨΗΣ

Στο σχήμα 6 που δίνεται παρακάτω, η μια εκ των δυο εντολών άλματος επιστρέφει τον έλεγχο σε προηγούμενη εντολή, οπότε εδώ θα χρησιμοποιήσουμε επανάληψη. Αυτό που θέλουμε είναι η πρώτη εντολή άλματος να μην οδηγεί σε εντολή μέσα στην επανάληψη ώστε να απλοποιήσουμε τα πράγματα. Έτσι, αντί να χρησιμοποιήσουμε τη δομή *Μέχρις_ότου* θα χρησιμοποιήσουμε την *Όσο...επανάλαβε*. Στο σχήμα.7 δείχνουμε πως μετατρέπουμε μια δομή *Μέχρις_ότου* σε ισοδύναμη δομή *Όσο...επανάλαβε*. Η αντιγραφή της ομάδας εντολών της επανάληψης έξω από αυτήν είναι επιτακτική έτσι ώστε το μπλοκ αυτό να εκτελείται τουλάχιστον μια φορά.



Σχήμα 6. *Ένα πιο σύνθετο παράδειγμα με άλμα σε προηγούμενη εντολή.*

Αρχή_Επανάληψης Εντολή_2 **Μέχρις_ότου** (Συνθήκη=Ψευδής)	Εντολή_2 **Όσο** (Συνθήκη=Ψευδής) **επανάλαβε** Εντολή_2 **Τέλος_επανάληψης**

Σχήμα 7. *Ισοδυναμία μεταξύ των δυο δομών επαναλήψεων*

Οπότε χρησιμοποιώντας τη δομή Όσο...επανάλαβε παίρνουμε το παρακάτω ισοδύναμο πρόγραμμα. Η απαλοιφή και της άλλης εντολής goto είναι πλέον απλή υπόθεση αφού αναλύθηκε στην προηγούμενη ενότητα. Έτσι προκύπτει το τελικό πρόγραμμα όπως φαίνεται στο παραπάνω σχήμα.

Αν (Συνθήκη1=Αληθής) **τότε** **GOTO** E3 Εντολή_1 Εντολή_2 E3: Εντολή_3 **Όσο** (Συνθήκη2=Ψευδής) **επανάλαβε** Εντολή_1 Εντολή_2 Εντολή_3 **Τέλος_επανάληψης**	**Αν** (Συνθήκη1=Ψευδής) **τότε** Εντολή_1 Εντολή_2 **Τέλος_αν** E3: Εντολή_3 **Όσο** (Συνθήκη2=Ψευδής) **επανάλαβε** Εντολή_1 Εντολή_2 Εντολή_3 **Τέλος_επανάληψης**

Σχήμα 8. *Αριστερά φαίνεται το πρόγραμμα που προκύπτει μετά τον πρώτο μετασχηματισμό και δεξιά φαίνεται το δομημένο πρόγραμμα μετά την απαλοιφή και της δεύτερης εντολής goto.*

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην εργασία αυτή παρουσιάσαμε μια πρόταση διδασκαλίας για τη μετατροπή ενός προγράμματος/αλγορίθμου με εντολές άλματος στο δομημένο ισοδύναμό του. Ισχυριζόμαστε ότι ο

μετασχηματισμός αυτός έχει σημαντική διδακτική αξία στα πλαίσια ενός εισαγωγικού μαθήματος στον προγραμματισμό. Αποτελεί έναν ακόμα εναλλακτικό τρόπο για να κατανοήσει ο μαθητής τον τρόπο λειτουργίας των τριών δομών ακολουθίας, επιλογής και επανάληψης του δομημένου προγραμματισμού. Χρησιμοποιήσαμε την εμπειρία που έχουμε από το μάθημα Ανάπτυξη Εφαρμογών της Γ' Λυκείου, για να βρούμε έναν τρόπο παρουσίασης που να είναι κατανοητός στους μαθητές. Καταλήξαμε στη διαγραμματική αναπαράσταση όπου φαίνεται καλύτερα η αλλαγή στην ροή του προγράμματος, σαν συνέπεια μιας εντολής άλματος. Τέλος πιστεύουμε ότι αξίζει μια αναφορά στην εντολή goto όχι μόνο για ιστορικούς λόγους, αλλά και επειδή οι δομές επιλογής και επανάληψης υλοποιούνται σε χαμηλό επίπεδο από την εντολή άλματος JMP.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. ACM, 11:147-148.
2. Bohm C. and Jacopini G. (1966), Flow diagrams, Turing machines and languages with only two formation rules, Communications of the ACM, 9(5):366-371.
3. Dijkstra E.W. (1968) Go to statement considered harmful. Communications of the ACM, 11:147-148.
4. Scanlan D.A. (1989) Structured flowcharts outperform pseudocode: An experimental comparison. IEEE Software, 6(5), 28-36.
5. Williams M.H. and Ossher H.L. (1978), Conversion of unstructured flow diagrams to structured. Comput. J, 21(2).
6. Zhang F. and D'Hollander E.H. (2004), Using Hammock Graphs to Structure Programs, IEEE Transactions on Software Engineering, Vol 30, No 4.
7. Βακάλη Α. et. al. (1999), Ανάπτυξη Εφαρμογών σε Προγραμματιστικό Περιβάλλον, ΥΠΕΠΘ, Παιδαγωγικό Ινστιτούτο, Αθήνα.